Evaluación y Optimización del proceso de reconocimiento de objetos con sensores de visión y profundidad

Alberto Sánchez Romero

Directores:

Ana Cristina Murillo Arnal
Eduardo Montijano Muñoz

Trabajo Fin de Máster
Máster en Ingeniería de Sistemas e Informática

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Diciembre 2013

Resumen

El reconocimiento automático de objetos es uno de los temas más populares y que más interés provocan en el campo de la visión artificial. Esto se debe principalmente a las múltiples aplicaciones que pueden beneficiarse de algoritmos capaces de reconocer la información presente en el entorno. Pueden ser desde aplicaciones en el ámbito de la industria, orientadas al control de la producción y la calidad, a aplicaciones en entornos domésticos como videojuegos o tareas de asistencia, sin olvidarse del ámbito de la robótica de servicio.

La actual aparición de sensores de bajo coste que combinan información de color y profundidad de la escena, conocidos como sensores RGB-depth o RGB-d, ha generado grandes oportunidades para el desarrollo de aplicaciones relacionadas con la detección automática de objetos. El hecho de incluir información de profundidad en la escena para cada píxel de la imagen resulta un complemento muy beneficioso para los algoritmos típicos de reconocimiento. Sin embargo, los algoritmos que existen actualmente para resolver este problema todavía se encuentran en una fase inicial de desarrollo, obteniendo resultados limitados y, en la mayor parte de los casos, siendo capaces únicamente de distinguir un número pequeño de objetos diferentes.

Este proyecto presenta un sistema completo de reconocimiento automático que es capaz de reconocer de manera eficiente un gran número de objetos. Para ello se hace uso de información 3D proporcionada por los sensores RGB-d y de técnicas de tratamiento de información 2D y 3D para poder realizar la clasificación de los objetos. El conjunto de datos que se ha empleado para la implementación se ha obtenido de una *dataset* de imágenes muy utilizada en la comunidad investigadora a nivel internacional.

Se parte de un estudio de la literatura relacionada con las técnicas básicas de visión por computador, técnicas para el manejo de la información obtenida de sensores RGB-d, sobre todo para representación de la información mediante nubes de puntos en 3D y el cálculo y uso de descriptores. También ha implicado al aprendizaje de la plataforma de desarrollo ROS, así como del sistema de reconocimiento del que se partía en este trabajo (un trabajo previo del grupo de investigación).

Este proyecto presenta una evaluación exhaustiva de dicho sistema base y propone una serie de mejoras, prácticas y teóricas, para aumentar el rendimiento, flexibilidad y formalización del mismo. Los experimentos realizados durante este proyecto demuestran las mejoras obtenidas tanto en coste computacional, como la gestión de recursos; así como las mejoras obtenidas por el algoritmo de clasificación. El resultado final es un sistema completo de reconocimiento, diseñado para el manejo de *dataset* de imágenes de gran tamaño y con capacidad de devolver resultados probabilísticos en función de cada objeto. Finalmente, se presentan las conclusiones y se proponen líneas de trabajo futuro.

Índice general

Indice II

1. Introducción **3**
1.1. Motivación y trabajo relacionado . . . 3
1.2. Objetivos y alcance . . . 4
1.3. Herramientas y entorno de trabajo . . . 5
1.4. Organización de la memoria . . . 5

2. Proceso de reconocimiento de objetos **7**
2.1. Representación de la información . . . 7
2.1.1. Nube de puntos . . . 7
2.2. Descriptores de imagen . . . 9
2.2.1. Introducción . . . 9
2.2.2. Tipos de descriptores . . . 9
2.2.3. Descriptores utilizados . . . 10
2.3. Pre-procesado de la escena . . . 15
2.3.1. Eliminar rango de visión . . . 15
2.3.2. Eliminar planos de fondo . . . 16
2.3.3. Clusterización de puntos pertenecientes al mismo objeto . . . 17
2.4. Clasificación de una imagen . . . 19
2.4.1. Fase de entrenamiento . . . 19
2.4.2. Fase de consulta . . . 20

3. Pruebas y Resultados **25**
3.1. Métodos de evaluación . . . 25
3.1.1. Matriz de confusión . . . 25
3.1.2. *Precision-Recall* . . . 26
3.2. Diseño de los experimentos . . . 27
3.2.1. Entorno de desarrollo . . . 27
3.2.2. *Dataset* de imágenes . . . 29
3.2.3. Base de datos de características . . . 30
3.2.4. Proceso de clasificación . . . 30
3.3. Análisis de resultados . . . 33
3.3.1. Coste computacional . . . 33
3.3.2. Evaluación de la calidad del sistema de reconocimiento base . . . 34

4. Conclusiones y Trabajo futuro **43**
4.1. Conclusiones . . . 43
4.2. Trabajo Futuro . . . 43

Bibliografía **45**

Anexos **46**

A. Sensor RGB-d Kinect **47**

B. Arquitectura del sistema de reconocimiento **49**

C. Información de los objetos del *dataset* **53**

D. Experimentos sobre el sistema de reconocimiento 3D **59**

E. Mejoras de rendimiento **65**

F. Creación de modelos de los objetos a reconocer **69**

Agradecimientos

Este proyecto no se podría haber realizado sin la ayuda de Ana Cris y Edu, a quienes agradezco la dedicación que me han prestado durante la realización del mismo. También quiero agradecerles la confianza depositada, permitiéndome trabajar en este proyecto que ha despertado mi interés por la investigación.

De igual manera quiero agradecerles a mis compañeros del CUD, en especial a Nuria, Etel, Tere y Marta, su tiempo. Y por último mis padres y a mi hermana, por estar siempre apoyándome.

Capítulo 1

Introducción

En el presente capítulo se describe la motivación que ha llevado al desarrollo de este proyecto, los objetivos y alcance del mismo. Así mismo, se describen las herramientas utilizadas y el entorno de trabajo, para finalmente comentar la organización de la memoria.

1.1. Motivación y trabajo relacionado

La comprensión de escenas 3D es un problema fundamental en percepción y robótica. El poder reconocer, modelizar y localizarse en el entorno es un requisito previo para otras tareas más complejas de interacción en robótica. Hasta la explosión de la visión por computador (gracias a procesadores mas potentes y a sensores de visión cada vez mejores y más baratos), la mayoría del trabajo en cuanto a localización en robótica se había centrado en el uso de sensores de rango, como escáneres láser o dispositivos sónar [1, 2].

Desde hace varios años, se han empezado a utilizar de forma habitual sensores de visión artificial para estas tareas de localización y construcción de modelos del entorno [3, 4, 5]. En cuanto al análisis y modelado del entorno con sensores de visión, y en particular en las escenas interiores, como habitaciones o despachos, uno de los retos que siguen activos es la detección y reconocimiento de objetos. Debido a la variación en la apariencia de los objetos, el tamaño, las oclusiones, la gran variedad de elementos que se pueden encontrar o las diversas escalas, entre otros factores.

En los últimos años, este problema de detección y análisis de escenas 3D ha avanzado mucho. La aparición de sensores RGB-depth, de bajo coste como el sensor Kinect[1] o el sensor Asus[2], que brindan la posibilidad de obtener información combinando color y profundidad, han favorecido el desarrollo de características y algoritmos para trabajar directamente con información 3D; y de esta forma se ha mejorado notablemente la precisión en la detección de objetos.

Es precisamente el reconocimiento de objetos con sensores RGB-d el tema general de este proyecto. La idea es poder desarrollar un sistema de visión capaz de detectar y clasificar ciertos objetos de uso cotidiano para su posterior integración en un sistema robótico que necesite interactuar con ellos, por lo que este proyecto se enmarca en el campo de la visión por computador aplicada a la robótica.

Encontramos un amplio abanico de investigaciones relacionadas con el reconocimiento de objetos, atendiendo a diferentes parámetros o características del objeto como pueden ser sus cualidades físicas color, forma, material. Por ejemplo en [6] encontramos una propuesta para reconocimiento de objetos que necesita información combinada de varias vistas del objeto a reconocer. Además de utilizar la información de apariencia de las imágenes, se puede añadir el uso del lenguaje natural para mejorarlo [7]; aunque precise seleccionar un gran vocabulario de palabras clave, y siendo muy dependiente del idioma.

En general, todo sistema automático de reconocimiento requiere de un conocimiento a priori del entorno, por parte del sistema, que debe poder interpretar imágenes y reconocer objetos. Para

[1] www.xbox.com/es-ES/Kinect
[2] www.asus.com/Multimedia/XtionPROLIVE

ello, es necesario entrenar o construir un modelo de referencia para nuestro sistema a partir de *datasets* de imágenes que reúnen información de objetos de ejemplo capturados, en nuestro caso con sensores RGB-d. En los *dataset* de imágenes que vamos a utilizar en este proyecto, la información de referencia de cada imagen esta compuesta por el conjunto de objetos que aparecen en la imagen y su segmentación. [8]

El objetivo de esta fase de aprendizaje previa es construir un sistema de clasificación o reconocedor, que dada una nueva imagen, nos clasifique lo que ve, entre los posibles objetos que el sistema conoce. Los tipos de clasificadores con los que podemos trabajar pueden ser **discriminativos** o **generativos** [9]. En las técnicas típicas del enfoque discriminativo como *Adaboost* [10], *SVM* (Support Vector Machines) [11] se necesita un entrenamiento para crear un espacio de muestras o modelado. La atención se centra en aprender los límites de las muestras, sin tener en cuenta la densidad de las mismas. Para una nueva consulta, se analiza como está dividida la información del espacio de muestras, y se intenta dividir hasta acotar la más cercana. Mientras que en un enfoque generativo por ejemplo un clasificador *Naive Bayes* [12] se trabaja con un espacio de muestras, en el cual se construye un modelo a partir de un conocimiento a priori de unos pocos ejemplos, y dada una nueva muestra, se trata de encontrar que modelo que es más probable que la haya generado.

Este trabajo propone una mejora sobre el sistema de reconocimiento, al que denominaremos sistema de reconocimiento base [13], desarrollado anteriormente en el grupo de investigación donde se desarrolla este proyecto, y una evaluación formal y exhaustiva con *datasets* de imágenes estándar utilizados para el reconocimiento de objetos con cámaras RGB-d. Una de las diferencias principales con respecto al sistema base es que el sistema diseñado en este trabajo está pensado para el manejo de *datasets* de imágenes de gran tamaño y con capacidad de devolver resultados probabilísticos como resultado de la clasificación (en lugar de respuestas binarias), como se detalla más adelante en el Capítulo 3. Esto se debe a que al introducir una cantidad mucho mayor de objetos, tanto los usados como referencia como los que hay que evaluar, hay mayor variedad y también más posibilidades de confusión de unas clases con otras, por lo que es más fácil equivocarse. En vez de tomar decisiones únicas sobre el objeto que se encuentra en la imagen a evaluar, se devuelve una lista de objetos más probables, con las probabilidades asociadas a cada uno de ellos. Al emplear un volumen de información mucho mayor, son necesarias una serie de mejoras en el diseño e implementación del sistema, debido a las restricciones hardware y a la búsqueda de eficiencia. Por ello se ha estudiado que cantidad mínima de referencia es necesaria.

Para la realización de nuestro sistema se ha optado como modelo de clasificación un enfoque generativo. Esto es debido a que los sistemas discriminativos básicos requieren un aprendizaje o entrenamiento, para todos los objetos, cada vez que se requiera añadir una nueva clase u objeto. Sin embargo, con un enfoque generativo, el poder añadir un nuevo objeto al sistema solo requiere tener unas cuentas imágenes de ejemplo de dicho objeto, y extraer los datos necesarios de estos ejemplos.

1.2. Objetivos y alcance

Este proyecto tiene como principal objetivo desarrollar un sistema de reconocimiento de objetos mejorado, que permita manejar grandes cantidades de información, dotando al sistema de una mayor capacidad de aprendizaje, así como de un sistema de clasificación con una mayor robustez a la hora de clasificar objetos dentro de escenas. Tras el análisis del sistema de reconocimiento base, desarrollado recientemente en el grupo de investigación [3], se lleva a cabo un proceso de refinamiento con el propósito de mejorar los resultados iniciales y adaptarlo a los nuevos requisitos, así como una evaluación exhaustiva comparando con el estado del arte en reconocimiento de objetos con cámaras RGB-d.

Para poder cumplir estos objetivos se procede, como primer paso, a decidir que *dataset* de imágenes público emplear durante el proyecto. Al mismo tiempo se estudian las técnicas básicas de visión por computador utilizadas en el trabajo de base [13] para las tareas de reconocimiento visual. Estas técnicas incluyen técnicas de segmentación de imagen, descriptores y características de imágenes así como algoritmos de comparación para las mismas. Una vez revisada la literatura, se

[3] robots.unizar.es

selecciona la plataforma de desarrollo, el driver de interacción con el sensor RGB-d, y las librerías para el procesamiento de imágenes. A esto se le añaden otra serie de criterios como son, la usabilidad en aplicaciones reales, complejidad, integración, y la posibilidad de combinación entre ellos.

Una vez seleccionados, se procede a la implementación del sistema y a su refinamiento progresivo con el propósito de obtener mejores resultados, para finalmente, mediante sistemas probabilísticos, elevar su robustez. Como paso final se realiza un estudio exhaustivo de los resultados obtenidos así como una elaboración de las conclusiones, al mismo tiempo que se proponen nuevas mejoras del sistema y líneas de trabajo futuro.

1.3. Herramientas y entorno de trabajo

Todo el desarrollo del proyecto se ha llevado a cabo bajo el sistema operativo Ubuntu versión 10.04 y se ha implementado utilizando el lenguaje C++. En una capa superior al sistema operativo, se ha utilizado el pseudo-sistema operativo ROS [4] (Robotic Operative System), sobre el cual ha sido desarrollado el sistema de reconocimiento. ROS nos ha facilitado el tratamiento de la información por su versatilidad y fácil integración con el sensor RGB-d empleado, que en nuestro caso ha sido el sensor Kinect. En pos de futuras líneas de investigación, se ha optado por el uso de la plataforma ROS, en lugar de desarrollar el sistema de reconocimiento de forma independiente de esta plataforma. Esto se debe a que se trata de una plataforma muy extendida en el ámbito investigador, ya que el uso de dicha plataforma puede favorecer la diseminación de esta propuesta en el mundo académico, así como la futura integración de este sistema de reconocimiento en una plataforma robótica.

Para posibilitar la interacción con el sensor Kinect, cuyas especificaciones se muestran en el Anexo A, se ha utilizado el driver OpenNI, que es uno de los más utilizados debido a que ofrece muchas funcionalidades. Este driver nos va permite leer y modificar la información que el sensor recibe de su entorno. Además para el procesamiento de imágenes y técnicas 2D se han utilizado diversos algoritmos y se ha hecho uso de librerías de visión por computador como OpenCV, que la podemos encontrar integrada en ROS. Mientras que para la gestión de información 3D se he empleado la librería PCL (Point Cloud Library) [14]. Ambas se encuentran muy presentes en desarrollo de aplicaciones en visión por computador, para el procesamiento de imágenes 3D.

1.4. Organización de la memoria

El resto de la memoria del proyecto se estructura en una serie de capítulos cuya idea principal y contenidos se exponen a continuación:

En el capítulo 2 se expone el proceso de reconocimiento de objetos empleado. Es aquí donde se describe como se representa la información, se define el concepto de descriptores de imagen, se presentan distintas clasificaciones de los descriptores y se hace referencia a los empleados en el sistema. A continuación se explica el proceso de segmentación de la escena que lleva a cabo el sistema; finalizando con una explicación de las fases de que consta la clasificación de una imagen.

El capítulo 3 detalla los métodos de evaluación adoptados para obtener los resultados. El diseño de los experimentos, la realización de los mismos y los resultados obtenidos son la parte central de este capítulo. Finalmente se ha llevado a cabo un análisis de resultados con el objetivo de elaborar conclusiones objetivas y proponer posibles mejoras en la respuesta del sistema.

El capítulo 4 contiene las conclusiones extraídas del trabajo y posibles líneas de trabajo futuro a las que este proyecto puede dar paso.

Las referencias consultadas para la elaboración de este proyecto pueden encontrarse al final de la memoria en la sección Bibliografía, seguidas por una serie de Anexos que detallan la información adicional sobre el proyecto: una descripción de la arquitectura del sistema, información de los objetos que conforman el *dataset* de imágenes, más resultados de los experimentos, las mejoras llevadas a cabo y una descripción de la creación de los modelos para cada objeto.

[4] wiki.ros.org

Capítulo 2

Proceso de reconocimiento de objetos

En este capítulo se explica todo el proceso desarrollado para reconocer de manera automática objetos en imágenes capturadas por una cámara RGB-d. Este proceso de reconocimiento lo podemos dividir en dos grandes etapas:

Una primera etapa, en la cual se procede a crear una base de datos de características representativas a partir de un conjunto de imágenes de ejemplo de los objetos a reconocer. Esta etapa consiste en la extracción y organización de dichas características.

La segunda etapa, engloba todos los pasos del procesamiento de información de una nueva imagen en la que queremos identificar que objetos aparecen. Esta etapa empieza, al igual que la etapa anterior, con la adquisición de una imagen RGB-d y el cálculo de la nube de puntos asociada a dicha imagen. Sin embargo, lo más normal es que esta imagen, que dispone de información de una escena general, pueda contener varios objetos e información no relevante para sistema. Por lo cual es muy importante realizar un buen pre-procesado de la imagen antes de pasar al proceso de reconocimiento.

En el resto de este capítulo se detalla como representar la información contenida en una imagen RGB-d mediante nubes de puntos en la Sección 2.1. Para, a continuación, en la Sección 2.2 explicar los descriptores de imagen utilizados, que facilitan la tarea de reconocimiento. En la Sección 2.3 se explican las técnicas de pre-procesamiento empleadas para, partiendo de una escena cualquiera, adquirida mediante una cámara RGB-d, identificar las regiones candidatas a contener un objeto. Finalmente, el proceso de reconocimiento del objeto se detalla en la Sección 2.4.

2.1. Representación de la información

Como se ha indicado, en este proyecto se ha trabajado con sensores RGB-d. Estos sensores son capaces de capturar, además de la imagen típica de color RGB, una segunda imagen de profundidad con la distancia expresada en metros asociada a cada píxel, en la Figura 2.1 se puede ver la imagen de color RGB y la de profundidad asociada. Y es el hecho de poder añadir una tercera dimensión para representar la información con la que se trabaja, la que implica que ya no es suficiente con una matriz de píxeles, que representan el color de cada imagen. Para ajustarse a estas necesidades se ha optado por trabajar con el modelo denominado nube de puntos que se explica a continuación.

2.1.1. Nube de puntos

Una nube de puntos se define como un conjunto de puntos en tres dimensiones, cada uno de ellos con un color asociado, obtenidos a partir de la información contenida en las imágenes de color y profundidad, como se puede ver en la Figura 2.1, la nube de puntos generada a partir de la imagen RGB y *depth* que podemos ver en la Figura 2.2.

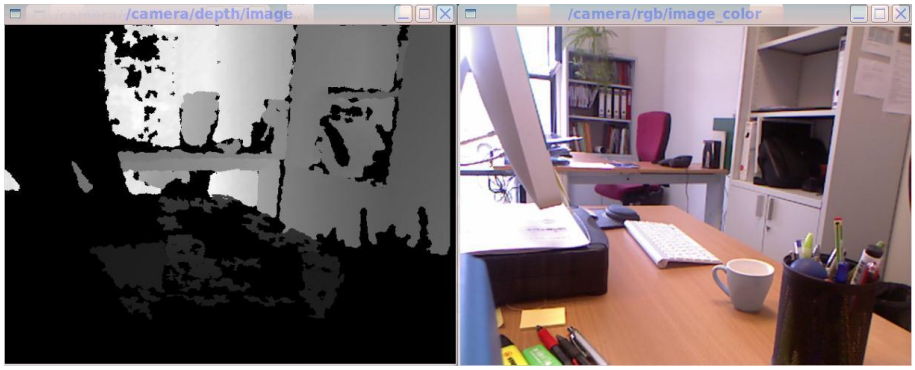


Figura 2.1: Ejemplo de imagen RGB (izquierda) e imagen de profundidad asociada (derecha). Cuanto más claro es el valor de la imagen de profundidad más lejos se encuentra el objeto representado en ese píxel.

Si se quiere conseguir los valores de un píxel, en este nuevo modelo, expresado en coordenadas de la imagen XY es relativamente sencillo si se cuenta con un mapa de profundidad que indique en metros la distancia de cada píxel a la cámara. Concretamente, para calcular los valores XY de un elemento $[i, j]$ del mapa de profundidad basta con aplicar las siguientes formulas:

$$\begin{aligned} X &= \frac{(i - c_i) \cdot \text{depth}[i,j]}{f} \\ Y &= \frac{(j - c_j) \cdot \text{depth}[i,j]}{f} \\ Z &= \text{depth}[i,j] \end{aligned} \tag{2.1}$$

Siendo c_i y c_j los índices del centro de la matriz de profundidad (en caso de que la matriz fuera de 640x480 elementos, los valores serían 320 y 240 respectivamente), $depth[i, j]$ el valor de la profundidad en la posición $[i, j]$ y f la distancia focal, un parámetro propio de la cámara con la que se tome la escena. Adicionalmente, a esta información se le puede añadir los valores de color que correspondan al elemento $[i, j]$ del mapa de profundidad.

De esta manera cada punto 3D de la escena cuenta con sus coordenadas XYZ y tiene asociado el color del píxel correspondiente a la imagen RGB.

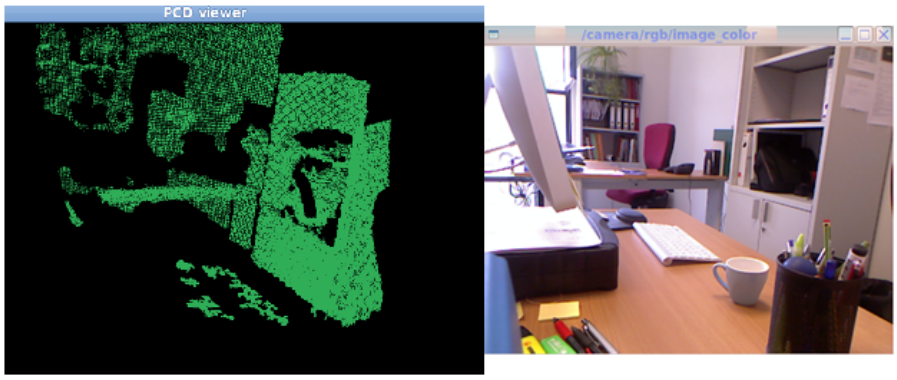


Figura 2.2: Ejemplo de imagen RGB (derecha) y la correspondiente nube de puntos (izquierda) obtenida. Se pueden aprecier ciertas formas de los objetos que componen la escena desde este punto de vista

2.2. Descriptores de imagen

2.2.1. Introducción

Aunque consideremos que una nube de puntos contiene un único objeto, no es conveniente trabajar con toda la información en crudo de cada punto de la nube. Por eso es muy recomendable realizar un procesamiento previo que extraiga la información relevante, para que esta pueda ser procesada de manera eficiente. Esto nos va a permitir trabajar siempre con una cantidad limitada y fija de datos. Todo este proceso en el que los datos de entrada se reducen para su posterior utilización se puede ver como una reducción de la dimensionalidad.

El contenido de una nube de puntos está codificado digitalmente en el valor de cada una de las unidades mínimas de información que la componen llamadas píxeles. De esta manera los píxeles representan el nexo de unión entre el contenido abstracto de sus valores y las propiedades de una imagen que entendemos como relevantes para las personas. Una propiedad es una característica que tiene un objeto, que permite identificarlo y compararlo con otros a través del establecimiento de semejanzas o diferencias. Un ejemplo de propiedad es color rojo, otro forma triangular, o ser un objeto de cocina. Las palabras como característica o atributo suelen utilizarse también para designar a las propiedades.

Puede pensarse que es mejor tener un número de características elevado para poder hacer una mejor diferenciación, esto es habitualmente falso. Aunque es cierto que incrementando el número de atributos (color, forma, textura, etc), la precisión con la que las variables de entrada pueden ser especificadas aumenta, este aumento también conlleva un crecimiento exponencial de la cantidad de datos que se necesitan para especificar el conjunto de objetos a reconocer. Este fenómeno se denomina maldición de la dimensionalidad [15].

Podemos destacar dos procedimientos para reducir la cantidad de información contenida en la nube de puntos. El primero consiste en la selección de atributos, donde simplemente se eligen las características más interesantes, desechando el resto. El segundo se basa en la transformación de características, donde se crea un nuevo espacio de características a partir de las variables originales. Debido a esto, este segundo procedimiento se denomina ***extracción de las características***. El uso de un conjunto de características limitado simplifica tanto la representación de los patrones como la complejidad del clasificador, lo que hace que el proceso de reconocimiento posterior sea más rápido y utilice menos memoria.

Como respuesta a la necesidad de describir el contenido de la información de una imagen o de una nube de puntos de forma objetiva y automatizada, surgen los descriptores de imagen. Un descriptor, como por ejemplo el color, es un conjunto de propiedades relacionadas entre si. Estas propiedades son los distintos colores: rojo, azul, amarillo, verde, etc. Puede decirse que los descriptores son las variables o aspectos observables en los objetos que pueden tomar distintos valores y las propiedades o características son cada uno de los distintos valores que puede tomar la variable.

2.2.2. Tipos de descriptores

Existen diferentes tipos de descriptores de imagen dependiendo del nivel de abstracción de la representación. Es posible clasificarlos en dos grandes grupos:

- **Descriptores de información general**: engloban descriptores también llamados de bajo nivel, que proporcionan una descripción respecto de la forma, color, regiones, textura y movimientos presentes en la imagen.
- **Descriptores de información de dominio especifico**: también llamados descriptores semánticos, proporcionan información acerca de los objetos y eventos que constituyen la escena.

Dentro de los descriptores de información general, podemos clasificar a los mismos según el nivel sobre el que actúan, es decir, sobre que regiones de la imagen realizan las distintas operaciones para generar los resultados que componen el descriptor.

- **Descriptores globales**: resumen el contenido de la imagen en un único vector o matriz de características. Poseen la ventaja de encapsular una gran cantidad de información de la imagen requiriendo un pequeña cantidad de datos para describirla. A pesar de su simplicidad, este tipo de descriptores han resultado ser ampliamente utilizados para diferentes tareas debido entre otras cosas a su bajo coste computacional unido a unas prestaciones relativamente buenas. Un representante de esta clase es el Histograma de color.

- **Descriptores locales**: son utilizados en aquellas tareas en las que una descripción local del contenido de la imagen resulta más apropiado. Actúan sobre regiones de interés, previamente calculadas o identificadas, construyendo un vector de características de esa región que tiene en cuenta la información contenida tanto en el punto de interés como en la región adyacente al mismo. Normalmente las regiones descritas se conocen como puntos de interés, también llamados puntos destacados o *keypoints*. Sin embargo estas regiones suelen referirse a bordes o pequeñas partes de la imagen. El descriptor entonces, está construido por la totalidad de los vectores de características calculados. A modo de ejemplo podemos mencionar el descriptor local SIFT.

Es necesario mencionar que existen diferentes clasificaciones de descriptores, y sin ánimo de ser exhaustivos y solamente a modo de ejemplo, se ha elegido la clasificación mencionada anteriormente debido a que abarca distintos tipos de descriptores exponiendo las diferentes categorías o herramientas de descripción en las que se dividen los descriptores de bajo nivel respecto de las características de la imagen sobre las que actúan. A continuación se detallan los descriptores de imagen elegidos para este proyecto.

2.2.3. Descriptores utilizados

En este proyecto, después de estudiar las opciones disponibles en la literatura y las librerías de manejo de Kinect, se ha optado por utilizar el descriptor de forma VFH (Viewpoint Feauture Histrogram) como descriptor principal o discriminante. Para a continuación apoyarnos en otros descriptores que se complementan mutuamente, el descriptor de textura SURF y el Histograma de color.

- **Viewpoint Feature Histogram o VFH**

  El primero y principal de los descriptores seleccionados para nuestro sistema. Se trata de un descriptor de datos para nubes de puntos 3D, que codifica geometría y punto de vista. Es decir, representa la información de la nube de puntos de forma más compacta, logrando capturar sus características más discriminantes. Este descriptor encaja dentro de los descriptores globales. En concreto, es un histograma calculado por cada imagen, que captura tanto la forma del objeto como el punto de vista desde donde se toma la imagen. Entre las razones encontradas para utilizar este descriptor, resaltan los buenos resultados que muestran sistemas de reconocimiento estudiados como [16, 17] que utilizan dicho descriptor. Además, al tratarse de un descriptor global, la memoria necesaria para almacenarlos es muy reducida, pues solo se calcula un descriptor por objeto, y las operaciones de comparación entre ellos son rápidas. No obstante, para almacenar el modelo de un objeto en la base de datos de características de forma efectiva, es necesario almacenar varios de estos descriptores desde distintos puntos de vista, y de esta manera tener la forma del objeto desde varias perspectivas.

  Para calcular este descriptor, primero es necesario calcular las componentes normales de la nube de puntos, en nuestro caso del objeto. Estas componentes se utilizan en gran variedad de áreas y existen numerosos métodos para calcularlas. Dada una superficie geométrica, suele ser trivial inferir la dirección de la normal de un punto en dicha superficie como el vector perpendicular a la superficie en dicho punto. Sin embargo, puesto que las nubes de puntos representan conjuntos de puntos en la superficie real es necesario usar aproximaciones para inferir las normales directamente desde el punto de la nube. Para calcular dichas normales se ha utilizado una función implementada en la librería PCL. Para más información de los cálculos utilizados se puede consultar [17]. En la Figura 2.3 se observa una representación de las normales de un subconjunto de los puntos de un objeto.

Figura 2.3: Representación de las normales de un objeto. El objeto, en este caso una jarra, se muestra como fondo y los vectores de color blanco representan la dirección de las normales en un subconjunto de los puntos de la nube capturada.

Una vez calculadas las componentes normales, se puede calcular el descriptor VFH. Para ello, se calculan los ángulos *pan-tilt-yaw* entre las normales de cada punto del objeto y las normales del centroide del objeto, para después plasmar la información en un histograma. Concretamente, para cada punto p_i del objeto, su componente normal n_i y el centroide p_c se calculan los siguientes ángulos:

$$\begin{aligned} \alpha &= v \cdot n_i \\ \phi &= u \cdot \frac{p_i - p_c}{d} \\ \theta &= \arctan(w \cdot n_i, u \cdot n_i) \end{aligned} \tag{2.2}$$

Donde u, v y w representan un marco de Darboux [1] elegido en p_i. La Figura 2.4 representa la selección del marco de Darboux y una representación gráfica de los tres ángulos calculados. A parte de los datos calculados hasta ahora, que describirían la forma del objeto, el descriptor incorpora información sobre el punto de vista desde el que se tomó la foto.

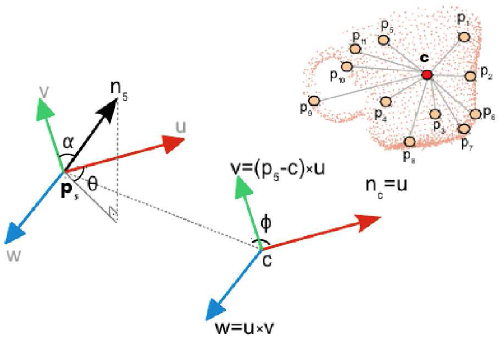


Figura 2.4: El descriptor almacena estadísticas de los ángulos relativos entre las normales de cada punto a la normal del centroide del objeto. La parte de abajo a la izquierda describe los tres ángulos calculados para un par de puntos de ejemplo.

[1] En geometría diferencial de superficies, un marco móvil construido en una superficie.

Esta información resulta útil para poder reconocer la orientación en la que se encuentra el objeto, aplicación interesante en campos como la robótica. Para obtener la información sobre el punto de vista se calcula un histograma de los ángulos que hacen cada componente normal con el punto de vista central trasladado a dicha normal. La Figura 2.5 representa gráficamente el concepto.

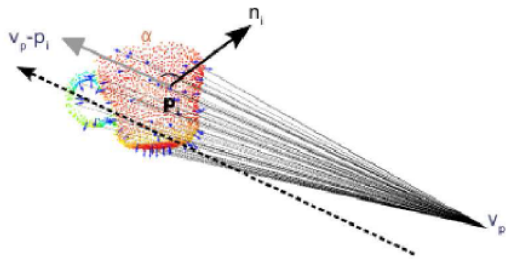


Figura 2.5: Parte de la información del descriptor VFH se calcula a partir de las estadísticas de los ángulos relativos del punto de vista central con cada componente normal de la nube.

Toda esta información se distribuye en un histograma. En la Figura 2.6 se muestra una representación de dicho histograma.

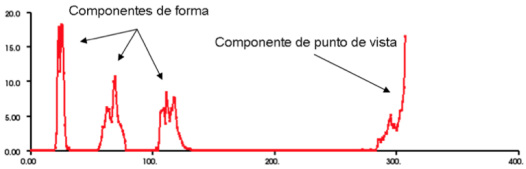


Figura 2.6: Ejemplo de un histograma VFH obtenido para un objeto. El histograma se distribuye o en 60 subdivisiones para cada uno de los tres ángulos *pan-tilt-yaw*, y 128 subdivisiones para la componente del punto de vista, lo que en total hace un total de 308 componentes.

En la siguiente Figura 2.7 podemos ver dos ejemplos de histograma VFH obtenidos para dos objetos diferentes.

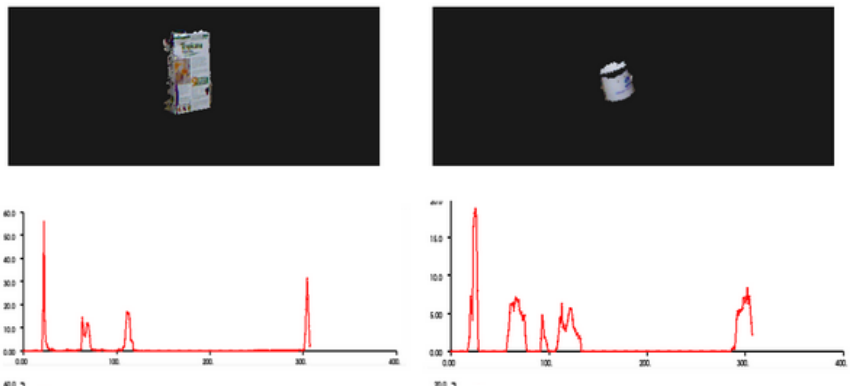


Figura 2.7: Histogramas VFH de dos de los objetos usados. Se pude apreciar tanto la diferencia en las componentes de forma, como de punto vista.

- **Speed-Up Robust Features o SURF**

  El siguiente descriptor que se ha decidido emplear es el SURF [18, 19], ya que se trata de uno de los más empleados y que mejor rendimiento han dado en aplicaciones de reconocimiento y visión por computador. Se trata de un descriptor que trabaja sobre información de los gradientes en 2D alrededor de ciertos puntos de interés de la imagen, es decir se engloba dentro de los descriptores locales.

  Este detector y descriptor surge como alternativa a los descriptores SIFT [20], descriptores muy robustos, invariantes a escala y orientación, pero a la vez demasiado complejos como para realizar aplicaciones de reconocimiento en tiempo real. La principal ventaja frente al descriptor SIFT es una mayor velocidad de calculo sin producirse perdidas en el rendimiento del descriptor. Estas mejoras se consiguen mediante la reducción de la dimensionalidad y la complejidad en el cálculo de los vectores de características de los puntos de interés obtenidos, mientras continúan siendo suficientemente característicos e igualmente repetitivos.

  Como ya hemos mencionado, fue desarrollado como un algoritmo capaz de detectar puntos característicos o de interés estables en una imagen. Estos puntos presentan cierta invarianza frente a diferentes transformaciones como translación, escala, rotación, iluminación y transacciones afines. En la Figura 2.8 puede verse el resultado de la extracción de puntos SURF de un objeto.

Figura 2.8: Puntos SURF encontrados en un objeto perteneciente al *dataset* de imágenes.

  La estabilidad de los puntos de interés es importante debido a que la comparación realizada entre objetos pertenecientes a dos imágenes diferentes se lleva a cabo mediante la comparación de los mismos puntos de interés. Sin embargo el problema más destacable de este tipo de descriptores es que solo son aplicables a objetos con textura. Además toda la robustez que presenta, tiene un precio, tanto en el coste computacional como en el tamaño del descriptor, aunque puede obtenerse con una mayor velocidad que los descriptores SIFT. Para subsanar alguno de estos inconvenientes, se ha optado por emplear de manera adicional un histograma de color, para buscar descriptores que se complementen entre si.

- **Histograma de color**

  Se ha empleado el histograma color [21], en adelante histograma, como alternativa para los casos en los que el reconocimiento tiene que abordar objetos que carecen de textura, en donde el descriptor SURF no pueden ayudarnos. Se trata de un descriptor global que trabaja sobre información 2D. Este histograma representa la frecuencia de aparición de cada una de las intensidades de color en la imagen. El histograma esta compuesto por diferentes rangos que representan un valor o conjuntos de valores de intensidad de color.

  El espacio color se define como un modelo de representación del color con respecto a los valores de intensidad. La dimensionalidad del espacio de color puede ser de una a cuatro dimensiones, siendo los espacio más representativos y utilizados los formados por tres componentes o canales de color. Para este proyecto, se consideró el espacio de color **RGB** (Red,Green,Blue), uno de los más utilizados para este tipo de tareas. El sistema RGB esta formado por los colores primarios Rojo Verde y Azul con valores entre $[0, 1]$, y cuya mezcla proporcionada resulta en el color deseado. El sistema RGB utiliza las coordenadas cartesianas como se

muestra en la Figura 2.9. Pero al tratarse de un sistema muy sensible a cambios de escena los histogramas obtenidos a partir de este espacio de colores son de poca utilidad.

Por lo tanto, se ha utilizado otro espacio de color que es algo más robusto a cambios de iluminación, el sistema HSV. El sistema **HSV** [22] (Hue, Saturation y Value) esta formado por estas componentes, que se muestran en la Figura 2.9. La componente *Value* representa la intensidad del color o brillo, la componente *Hue* representa lo que se conoce como tonalidad, y la componente de *Saturation* representa la densidad del propio color o la pureza. Siendo la componente Hue la que mayor peso tiene de las tres.

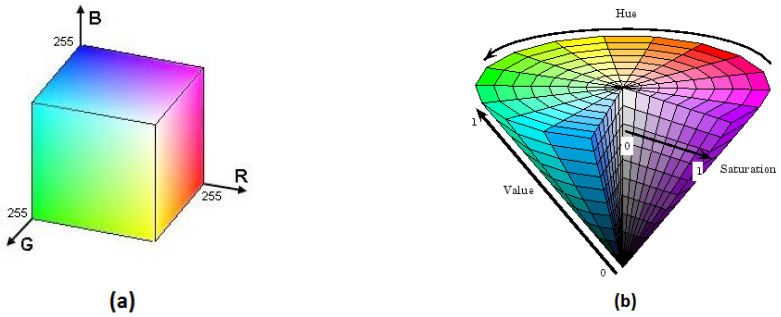


Figura 2.9: Espacios de color : (a) RGB y (b)HSV

La resolución de las distintas componentes no es uniforme, sino que se utiliza un mayor numero de bits para representar la componente *Hue* que para las dos restantes. Sin embargo todas las componentes varían en un rango también normalizado de entre $[0, 1]$. El espacio de color HSV guarda una mayor relación o esta más próximo a la manera que tienen el sistema de visión humano de percibir el color, que el espacio RGB.

Teniendo en cuenta que la descripción del color expuesta está formada por tres componentes, el histograma de color de una imagen, como descriptor, estará formado por la composición de los distintos histogramas de cada uno de los canales o componentes de color, construyendo así un único vector. A continuación podemos ver en la Figura 2.10 un representación del histograma en el espacio de color HSV.

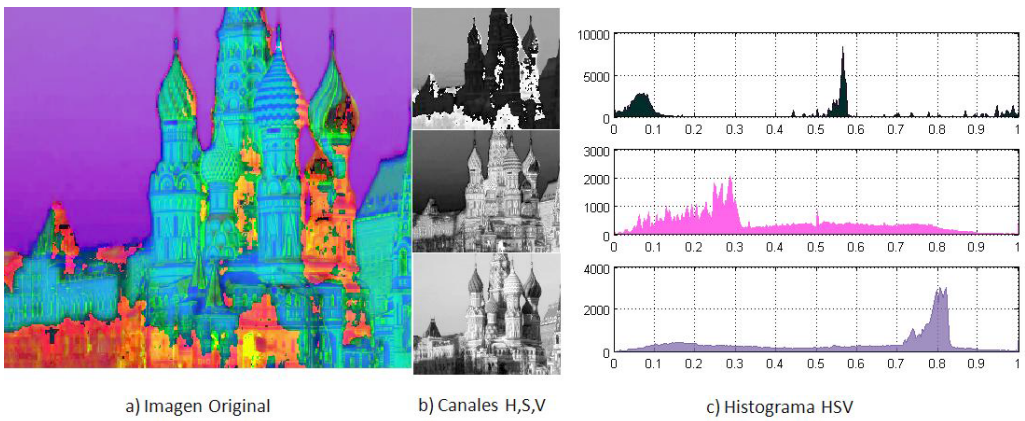


Figura 2.10: Representación del histograma HSV. a) representa la imagen original en el espacio de color HSV. b)Representación vertical de las tres componentes de color H,S,V de la imagen. c) representación del histograma de cada una de las componentes de la imagen.

El aspecto más atractivo y ventajoso del histograma es su simplicidad y velocidad de computación, tanto en la tarea de comparación como en la creación del descriptor. Resulta robusto frente a pequeños cambios de escala o pequeños movimientos de los elementos representados en la imagen y se muestra invariante respecto de la rotación sobre los ejes. Es sencillo y

compacto; presenta un bajo coste computacional, respecto del tamaño y tiempo de cálculo así como poca memoria necesaria para almacenarlo.

Sin embargo existen diversos inconvenientes asociados al mismo, como por ejemplo la falta de consideración de información espacial de las distribuciones de color. Se trata de un descriptor que no incluye información espacial: 2 imágenes completamente distintas pueden tener histogramas similares si tienen un numero similar de píxeles de cada color. Además las variaciones de iluminación pueden alterar el histograma de forma muy significativa.

Al tratarse de un descriptor global es menos representativo que los locales, por lo que el rendimiento que ofrecen a nivel requerido para identificar un objeto puede ser demasiado bajo, es por eso que, como en este trabajo, se suele utilizar en combinación con descriptores adicionales.

2.3. Pre-procesado de la escena

Es posible que no toda la información recibida del sensor sea útil para el sistema, ya que los objetos se encuentran sobre un fondo con ruido, por lo que es necesario realizar un pre-procesado de los datos antes de realizar la clasificación de los posibles objetos visibles. La etapa de reconocimiento dada una imagen de test comienza por tanto con el pre-procesado de la escena, la Figura 2.11 muestra un resumen, en el cual, se intenta eliminar la información no relevante contenida en la nube de puntos mediante tres pasos: el filtrado del rango de percepción del sensor, la substracción de planos dominantes y la clusterización de los puntos restantes. Una vez que se han obtenido las regiones con posible información relevante, se pasarán al algoritmo de clasificación. A continuación se detallan estos tres pasos de segmentación o pre-procesado de la nube de puntos. Es decir, esta es la etapa en la que se tratan los datos antes de proceder a su clasificación.

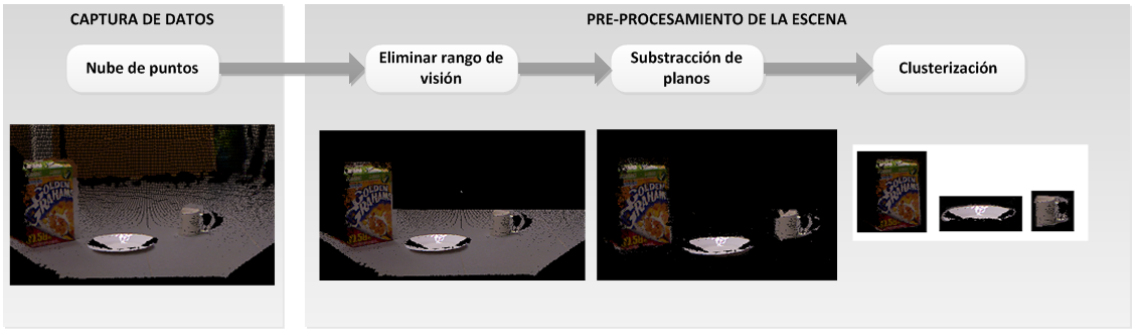


Figura 2.11: Proceso de pre-procesado de una nueva imagen. En la figura se pueden ver las diferentes etapas que atraviesa la nube de puntos, antes de poder obtener la información útil de cada posible objeto. El resultado al final de este proceso (*clusters* de píxeles contiguos), es lo que se le entrega al algoritmo de clasificación de objetos.

2.3.1. Eliminar rango de visión

El primer paso del pre-procesado consiste en la eliminación del rango de visión del sensor. El sensor Kinect tiene un rango de visión de alrededor 3,5 metros, pero solo vamos a considerar elementos a menos de 1.1 metros de distancia, ya que supondremos que lo que este más lejos, sera demasiado pequeño para ser reconocido o corresponde con elementos del entorno tipo puertas y paredes. Además, el sistema robótico final podrá "acercarse" a mirar más de cerca los elementos que considere. En la siguiente Figura 2.12, se muestra un ejemplo de la captura de una escena, y la eliminación del rango de visión del sensor.

Figura 2.12: Imagen que representa como queda la escena una vez eliminado el rango de visión del sensor. La imagen superior representa la nube de puntos original. En la imagen inferior la nube de puntos eliminando todo lo que se encuentra más allá de 1.1 metros.

2.3.2. Eliminar planos de fondo

El segundo paso de pre-procesado consiste en el cálculo de los puntos pertenecientes a superficies planas dominantes en la escena, para eliminarlos de nuestros puntos de interés (ya que corresponden a la estructura de la escena: paredes, mesa, suelo,....). Es decir, este paso consiste en una eliminación o substracción de los planos dominantes. La eliminación de planos suele realizarse mediante un algoritmo robusto, en nuestro caso el algoritmo *Random Sample Consensus* [23] o RANSAC. Se trata de un algoritmo que estima, dado un conjunto de datos, un modelo matemático deseado. En este proyecto, el modelo matemático buscado son los planos (suelo, mesa, pared), y el conjunto de datos, las nubes de puntos de los objetos detectados por el sensor. El principio sobre el que se basa el algoritmo es sencillo.

Es un algoritmo iterativo, en el que en cada iteración, se elige aleatoriamente un subconjunto de datos del conjunto total que formaran parte del hipotético modelo final. Después se comprueban los puntos restantes, para saber si encajan en el modelo. La estimación de las iteraciones restante cambia en cada iteración y significa el número de intentos necesarios para conseguir un conjunto en el que todos los datos pertenezcan al modelo con probabilidad K. En este paso los puntos calculados por el algoritmo RANSAC se eliminan de la nube de puntos final que contienen los objetos a clasificar. Respecto a las nubes de puntos, puede ocurrir que en el proceso de cálculo del plano, algunos de los puntos de la parte inferior de los objetos pueden ser tomados como parte del plano. Esto hace que en objetos relativamente planos como pueden ser los libros, cajas ... muchos puntos del objeto sean eliminados y como consecuencia no se pueda alcanzar el tamaño mínimo necesario para poder ser detectados. Continuando con la escena de ejemplo, en la Figura 2.13 se puede ver como queda la escena una vez eliminados los planos dominantes.

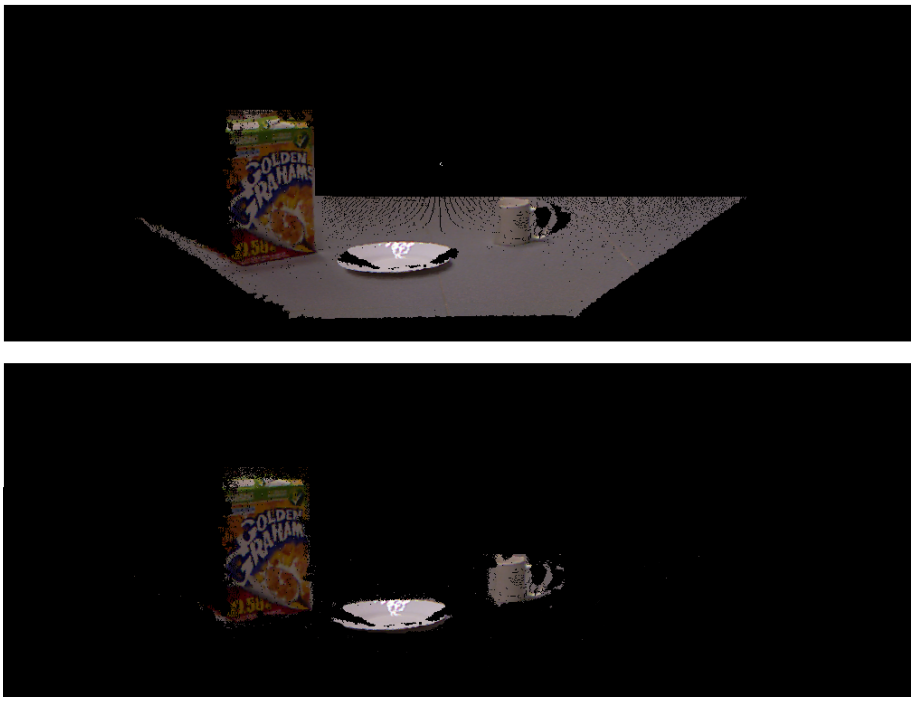


Figura 2.13: En la imagen superior podemos ver la imagen a la que habíamos eliminado el rango de visión. Debajo la misma imagen una vez eliminado el plano dominante, el suelo.

2.3.3. Clusterización de puntos pertenecientes al mismo objeto

El tercer paso del pre-procesado de la nube de puntos consiste en una "clusterización" de la nube de puntos para agrupar los puntos en grupos contiguos que son candidatos a representar posibles objetos. Si los pasos anteriores se han realizado de manera correcta, la nube de puntos solo contendrá conjuntos o grupos de píxeles aislados entre si que no formen un plano. Esta información aparentemente desordenada (cada uno de estos grupos tiene una etiqueta en común, que se ocupa de diferenciar entre los grupos de píxeles), contiene los objetos que se quieren reconocer, además de ruido u otros elementos de la escena, necesita ser ordenada en *clusters* y así poder reconocer cada objeto de forma separada.

Para lograr la agrupación de la información, se ha optado por utilizar un método de **extracción de *clusters* euclídeo**. Se trata de un método que divide el modelo de nube de puntos desorganizado P en partes más pequeñas para reducir el tiempo de procesamiento de P significativamente. La implementación del modelo se lleva a cabo mediante el uso de de una subdivisión del espacio 3D en una rejilla, utilizando cajas de un tamaño fijo, o algo más general, una estructura de datos *octree*[2]. Esta representación en particular es muy rápida de construir y es útil en situaciones en las que o bien se necesita una representación volumétrica del espacio ocupado, o los datos resultantes en cada caja 3D (u hoja *octree*) puede ser aproximada con una estructura diferente. En un sentido más general, se puede hacer uso de los vecinos más cercanos y poner en práctica una técnica de *clustering* que esencialmente es similar a un algoritmo de relleno [3].

Supongamos que tenemos una nube de puntos con una mesa y objetos sobre ella. Queremos encontrar y procesar los *clusters* individuales de los posibles objetos situados sobre el plano. Suponiendo que se utiliza un estructura *kd-tree*[4] para encontrar los vecinos más cercanos, los pasos del algoritmo serían:

[2]Un octree es una estructura de datos basada en árbol para la gestión de datos 3D dispersos. Cada nodo interno tiene exactamente ocho hijos

[3]Determina el área formada por elementos contiguos en una matriz multidimensional

[4]Un kd-tree es una estructura de datos utilizado en informática para la organización de un número determinado de puntos en un espacio de dimensiones k.

1. Crear una representación con un *Kd-tree* para la nube de puntos de entrada P;
2. Crear una lista de cluster C, y una cola de puntos que necesitan ser comprobados Q;
3. Después, para cada punto $p_i \in P$ realizar los siguientes pasos:
   - Añadir p_i a la cola actual Q;
   - Para cada punto $p_i \in Q$:
     - Buscar el conjunto de P_k^i de vecinos de p_i en una esfera con radio $r < d_{th}$;
     - Para cada vecino $p_i^k \in P_i^k$; comprobar si el punto ya ha sido procesado, y si no añadirlo a Q.
   - Cuando la lista de todos los puntos en Q se ha procesado, añadir Q a la lista de clusters C y limpiar la cola Q dejándola vacía
4. El algoritmo termina cuando todos los puntos $p_i \in P$ han sido procesados y son ahora parte de la lista de clusters de C.

Una vez acabado este proceso, es conveniente nuevamente, eliminar aquella información que no vaya a servirnos. De lo que se trata es de ignorar los *cluster* demasiado pequeños o demasiado grandes, los primeros, porque lo más probable es que sean producto del proceso de la eliminación de planos; mientras que los segundos sean de un tamaño superior a los objetos que queremos reconocer. En la Figura 2.14 se puede ver el resultado del proceso de clusterizacion.

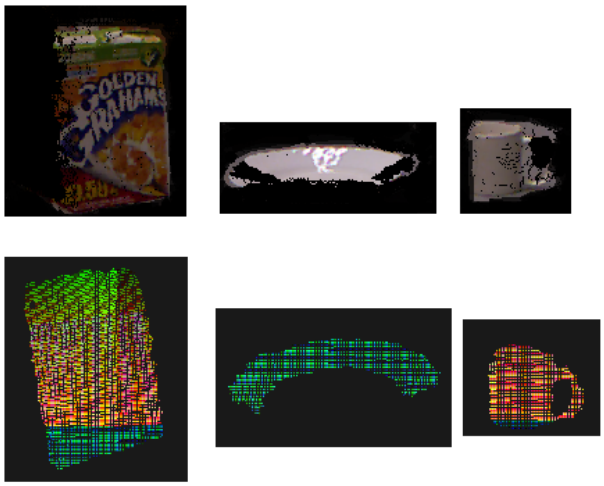


Figura 2.14: Objetos de la escena ya clusterizados. Debajo el resultado de agrupar la imagen, donde cada tonalidad representa un *cluster* distinto.

2.4. Clasificación de una imagen

Esta sección presenta las dos fases de las que se compone el método de clasificación, la fase de entrenamiento o aprendizaje y la fase de consulta o test. Cada fase se puede desglosar en varios apartados.

2.4.1. Fase de entrenamiento

En esta fase, el sistema es entrenado con diferentes instancias de cada objeto, que le permitan reconocer un objeto diferente de una misma clase más adelante durante la fase de consulta. En esta fase de entrenamiento se deben atender los siguientes aspectos:

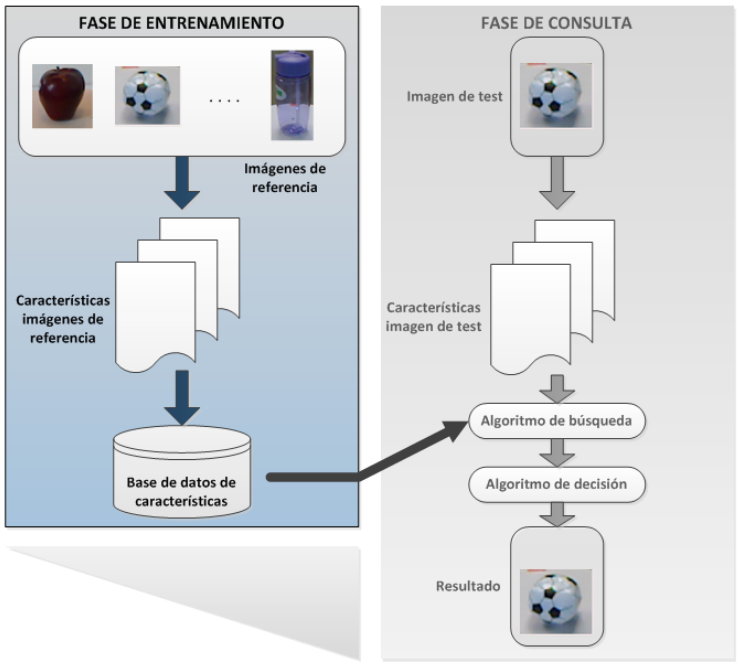


Figura 2.15: Fases en las que se divide el proceso de clasificación de una imagen. En esta imagen se resalta la fase de entrenamiento.

- **Recopilación de datos.** Hay que tener en cuenta cual va a ser el campo o criterio de clasificación, es decir, que tipo de clases se van reconocer y clasificar. Los *dataset* son muy diferentes en función de lo que se quiera clasificar: objetos, personas, palabras, etc. Estos presentan una organización estructurada en forma de árbol, donde cada rama del *dataset* es una clase fácil de reconocer y las hojas son los diferentes objetos, que a su vez tienen otras ramas que pueden contener instancias. En internet podemos encontrar varios *dataset* en función del campo de aplicación en el que queremos evaluar el clasificador. Por ejemplo Caltech 101[5], Warehouse google[6] incluso otra opción puede ser la creación de un *dataset* propio.

  Más adelante, en la sección 3.2.2, se describe el *dataset* de imágenes elegido, compuesto por una gran variedad de clases de objetos, que pueden verse en detalle en el Anexo C. En la Figura 2.15 este punto se corresponde con las imágenes de referencia.

[5] www.vision.caltech.edu
[6] sketchup.google.com/3dwarehouse.

- **Creación de los modelos y elección de la representación adecuada.** Una vez que se ha elegido qué datos se van a emplear, hay que considerar qué tipo de características pueden representar a una clase de manera correcta, o dicho de otra forma, cuáles van a ser los descriptores, para poder generar los modelos que aglutinen la información. Por ejemplo, si se necesita clasificar entre una manzana verde y otra roja, el color sería un descriptor acertado, mientras que si tenemos que discernir entre una silla y un vaso la forma o el tamaño sería lo más adecuado. Una vez que se ha seleccionado una serie de características habrá que diseñar o elegir que método de clasificación se va a utilizar.

  Como ya se ha explicado en la Sección 2.2.3 los modelos para cada uno de los objetos que utilizaremos estarán formados por un conjunto de descriptores (VFH, SURF, e Histograma de color). En el Anexo F se puede ver el proceso de creación de los modelos.

- **Entrenamiento del clasificador** Por último, una vez realizados los pasos anteriores, se selecciona un conjunto de ejemplos del *dataset*, que llamaremos de referencia o entrenamiento. Estos datos junto con sus descriptores o características correspondientes, se utilizan para construir el modelo de referencia, es decir, "entrenar" nuestro clasificador. Esta información se almacena para poder clasificar nuevos ejemplos en las diferentes clases de interés. A estos datos de "referencia" almacenados, los podemos denominar base de datos de características.

2.4.2. Fase de consulta

Concluida la fase de entrenamiento, se puede ejecutar y evaluar la fase de consulta del reconocedor. Por cada nueva imagen a reconocer, el sistema extrae sus descriptores o características, para luego evaluar la similitud respecto al modelo aprendido almacenado, decidiendo en nuestro caso que imagen de referencia presenta mayor similitud y asignando al test la clase de dicho ejemplo de referencia. Es decir, nuestro clasificador esta basado en un sistema "nearest neighbour".

En la Figura 2.16, en la fase de consulta, se pueden ver las dos partes más importantes de esta fase: el **Algoritmo de Búsqueda** (similitud) y el **Algoritmo de Decisión**.

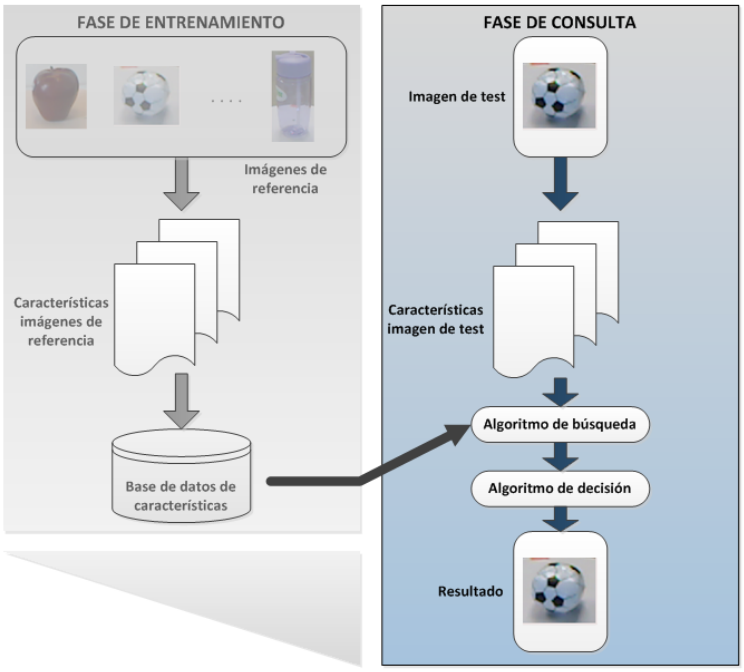


Figura 2.16: Fases en las que se divide el proceso de clasificación de una imagen según el "vecino mas cercano". En esta imagen se resalta la fase de consulta.

2.4.2.1 Evaluación de similaridad de la nueva imagen y nuestro modelo

Una vez que se ha decidido cómo va a representarse la información de las imágenes, el siguiente paso natural es el de decidir qué medida de similitud se va a utilizar. Elegir de forma adecuada la medida de similitud que mejor se ajuste a las necesidades del problema es de gran importancia, ya que el comportamiento del sistema puede variar dependiendo de la medida utilizada.

Búsqueda del vecino más cercano

Nuestro sistema de clasificación se basa en la búsqueda del vecino o vecinos más cercanos a la nueva imagen dentro de nuestra información de referencia. Por lo tanto, debemos comparar los descriptores de la imagen de test con todos los descriptores de las imágenes de referencia, Figura 2.17, obteniendo un número de correspondencias para cada imagen, según lo similares que sean los descriptores de cada una de ellas. La imagen de referencia con más correspondencias será considerada la más similar.

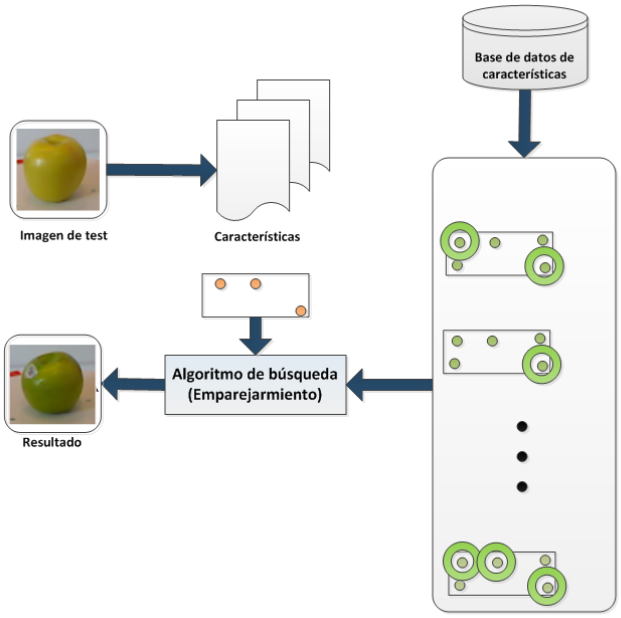


Figura 2.17: Representación del algoritmo de búsqueda del vecino más cercano. Las características de la imagen de test se comparan con las características de las imágenes de referencia, obteniendo como resultado aquella imagen de referencia donde se encuentren mas correspondencias con la imagen de test.

Se ha empleado para la búsqueda de correspondencias el algoritmo de **Búsqueda aproximada**; al presentar unos resultados aceptables [13]. La elección de este algoritmo implica que en la fase de entrenamiento se genere una estructura para ordenar los datos, los *kd-trees* [24]. La ventaja de este algoritmo es la disminución del coste de cómputo, aunque no puede garantizar encontrar la solución óptima.

Un ***kd-tree*** o k dimensional tree, es una estructura de datos de particionado del espacio que organiza los puntos en un espacio euclídeo de k dimensiones, los *kd-tree* son un caso especial de árboles BSP [7]. Se trata de un árbol de búsqueda binaria al que se le imponen unas limitaciones. Los *kd-tree* son muy útiles para las búsquedas de un rango o las búsquedas del vecino más cercano. En nuestro caso estamos trabajando con nubes de puntos tridimensionales, por lo que la dimensión

[7]Binary Space Partitioning es un método para subdividir de forma recursiva un espacio en conjuntos convexos por hiperplanos

de nuestros kd-trees será igual a tres, en la Figura 2.18 podemos ver una representación. Cada nivel del *kd-tree* se divide a su vez en hijos a lo largo de una dimensión especifica, utilizando un hiperplano que es perpendicular al eje correspondiente. En la raíz del árbol en la que se dividieron todos los hijos sobre la base de la primera dimensión (por ejemplo, si la primera coordenada de la dimensión es menor que la raíz será en el sub-árbol de la izquierda y si es mayor que la raíz, sera en el sub-árbol de la derecha). Cada nivel en el árbol se divide en la siguiente dimensión, volviendo a la primera dimensión cuando todos los demás se han agotado. Se repite el proceso en los sub-árboles de la izquierda y de la derecha hasta que el último árbol particionado este compuesto solo por un elemento.

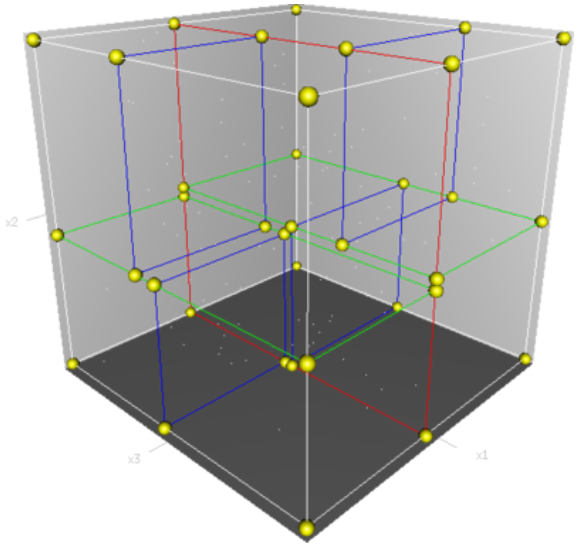


Figura 2.18: Representación de un *kd-tree* tridimensional y sus divisiones espaciales asociadas

Esta técnica sirve para comparar imágenes representadas tanto con descriptores globales como locales, la diferencia radica en cómo medir la distancia entre descriptores. A continuación se explica con más detalle cuales han sido las medidas utilizadas, dependiendo de la naturaleza del descriptor.

- **Búsqueda del vecino más cercano con descriptores locales**
  Para medir la distancia entre los descriptores locales, el SURF en nuestro caso, se ha aplicado la **distancia euclídea**. No se ha estudiado una medida más compleja ya que el número de descriptores a comparar puede ser muy grande, así como las componentes de cada descriptor, por lo que una medida más compleja resultaría en un tiempo de cómputo demasiado elevado. Esta medida puede obtener correspondencias incorrectas entre las imágenes, por lo que puede ser recomendable añadir un paso de estimación robusta con el objetivo de eliminar dichas correspondencias erróneas. El algoritmo **RANSAC** (*RAndom SAmple Consensus*) [23] es un buen método para conseguirlo. Ya que mediante la restricción geométrica que añade consigue rechazar las correspondencias que no sean consistentes con el modelo geométrico de la escena. Aunque aplicar el algoritmo RANSAC supone una importante mejora en los resultados basados en la búsqueda del vecino más cercano, no siempre es aconsejable utilizarlo, pues su coste de cómputo es elevado y supone un notable incremento en el tiempo de ejecución.

- **Búsqueda del vecino más cercano con descriptores globales**
  En el caso de utilizar los descriptores globales, como el número de comparaciones es menor, solo hay un descriptor por imagen que comparar, se ha evaluado el uso de una distancia más compleja y robusta que la euclídea, la técnica **Earth Mover's Distance**[25] (EMD).
  Conceptualmente, la distancia EMD se define como la cantidad de trabajo que llevara encajar la forma de un histograma en la del otro. Calcular esta distancia se basa en resolver el conocido problema de transporte Monge-Kantorovich [26]. Esta técnica se ha escogido para ser utilizada como medida de similitud entre los histogramas de color. Se ha elegido esta

medida porque el concepto se ajusta muy bien a las necesidades de los problemas, ya que la EMD es una medida de distancia entre histogramas, en vez de emparejamiento. Esto resulta muy útil ya que histogramas de color del mismo objeto pero en diferentes escenas pueden sufrir cambios de iluminación, lo que causa un desplazamiento de dichos histogramas, pero manteniendo la misma "forma".

2.4.2.2 Algoritmo de decisión: clase asignada a la nueva imagen

La creación de la lista de candidatos es un paso muy importante, pues hay que asegurar que en ella esté contenido el objeto al que corresponde el *cluster* que se está reconociendo, aunque debido a ello se introduzcan más candidatos. Los siguientes pasos que evalúan otros descriptores ya se encargarán de pulir la decisión. Por tanto, la lista de candidatos, definida como LC, contiene una serie de instancias similares al cluster que se está intentando reconer. La lista se crea de la siguiente manera: una vez extraído el descriptor VFH del *cluster* que estamos evaluando de la imagen de test, y calculada la distancia *Chi-cuadrado* entre este *cluster* y los ejemplos de la base de datos, se eligen los n vecinos más cercanos de la base de datos. De entre estos vecinos se incluyen en la lista todos los objetos cuya distancia d cumplan estas condiciones:

$$d < 3mindist \quad \text{si} \quad mindist \leq 20$$
$$d < 2mindist \wedge d < 100 \quad \text{si} \quad mindist > 20$$

Siendo *mindist* la distancia del primer vecino más cercano. Es muy posible que un mismo objeto de la base de datos corresponda a varios de los vecinos más cercanos que cumplen las condiciones arriba expuestas, pero solo conviene guardar una distancia por objeto en esta lista de candidatos. Por tanto, la distancia (x_i) para el descriptor VFH evaluado del objeto i de la lista de candidatos es la siguiente:

$$x_i = \left(\sum_{j=1}^{k} \frac{d_i^j}{k} \right) \cdot \left(1 - \frac{rep_i}{100} \right) \tag{2.3}$$

Siendo d_i^j la j-ésima mejor distancia *Chi-Cuadrado* del objeto i, rep_i el numero de veces que el objeto i satisface las condiciones para entrar en la lista de candidatos y k obtenido con la siguiente expresión:

$$k = \begin{cases} 3, & \text{si } rep_i \geq 3 \\ rep_i, & \text{si } rep_i < 3 \end{cases} \tag{2.4}$$

Esta distancia almacenada es la media de hasta las tres mejores distancias, pero se le resta un 1 % a esa distancia para cada vez que se repita el objeto, para premiar la repetición.

Siguiendo con el diagrama de funcionamiento del sistema, descrito en el Anexo B, una vez extraídas las características SURF de los elementos de la lista de candidatos, para los que resulten como objetos con textura, ya que se encontraron suficientes puntos SURF, se almacena la mejor medida de similitud. Esta medida de similitud entre dos imágenes Im_i y Im_j que utiliza el sistema viene dada por:

$$Sim = \frac{Matches_{ij}}{Max(NumDes_i, NumDes_j)} \tag{2.5}$$

Siendo $Matches_{ij}$ el número de correspondencias entre las imágenes i y j, y $NumDesc_i$ el número de descriptores extraídos la imagen i.

En una segunda etapa se calcula una medida de similitud global, normalizada entre cero y uno, que indica como de parecido es cada candidato al *cluster* que se está evaluando. Dado un candidato v, esta medida de similitud la definimos como $\text{Pond}(v) \rightarrow (0, 1)$.

Con todas las medidas calculadas, se procede a elegir el objeto y la clase que mejor representan al *cluster*. Esto se define formalmente mediante la siguiente ecuación:

$$\begin{aligned} O_{\text{pond}} &= O\left(\underset{v \in LC}{\text{máx}}\, \text{Pond}(v)\right), \\ C_{\text{pond}} &= C\left(\underset{v \in LC}{\text{máx}}\, \text{Pond}(v)\right). \end{aligned} \tag{2.6}$$

Donde $C(v)$ se define como la clase que representa la instancia v y $O(v)$ denota el objeto que representa la instancia v.

Denotamos mediante O_k al conjunto de instancias de la lista de candidatos que representan al objeto k,

$$O_k = \{v \in LC | O(v) = k\}, \tag{2.7}$$

y de manera analoga, C_k, representa el conjunto de objetos de la lista de candidatos que representan a la clase k,

$$C_k = \{v \in LC | C(v) = k\}. \tag{2.8}$$

Por ejemplo, si k se refiere a apple_1, O_k denota el conjunto de candidatos que son instancias de apple_1, mientras que C_k se refiere a la clase apple.

Teniendo esto en cuenta, la probabilidad de que el objeto de *query* se corresponda con el objeto k se define como:

$$P(k) = \frac{\sum_{v \in LC | O(v) = k} Pond(v)}{\sum_{v \in LC} Pond(v)}. \tag{2.9}$$

Capítulo 3

Pruebas y Resultados

Para determinar el rendimiento del sistema se han llevado a cabo una serie de experimentos que permitan conocer su comportamiento. En esta sección se van a presentar aquellos experimentos que mejor caracterizan el rendimiento del sistema.

3.1. Métodos de evaluación

En la clasificación de objetos, una imagen se clasifica de acuerdo a su contenido visual. Existen muchas métricas para conocer distintos aspectos de un sistema de clasificación. En este proyecto vamos a utilizar algunas de las métricas de evaluación más utilizadas en el ámbito científico para evaluar la calidad de un clasificador.

3.1.1. Matriz de confusión

La matriz de confusión se utiliza habitualmente en el aprendizaje supervisado para mostrar la relación entre las clases reales y las clases predichas por un sistema de clasificación. En la Figura 3.1 se muestra el esquema de representación de una matriz para una clasificación binaria:

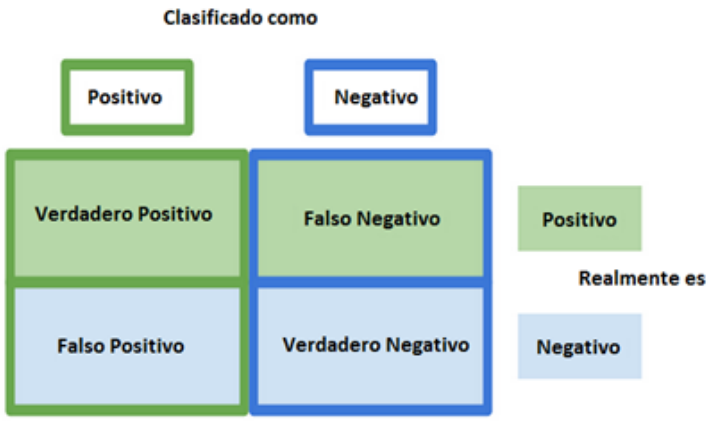


Figura 3.1: Tabla de confusión. Contiene la información acerca de las clasificaciones actuales y predicciones hechas por el sistema

Las filas representan la clase real de un objeto mientras que las columnas representan la clase predicha por el clasificador. En este contexto distinguimos cuatro posibles resultados de la clasificación:

Verdadero positivo (VP) es la predicción correcta de una muestra positiva; por ejemplo, un objeto que es una taza el clasificador lo ha identificado como una taza.

Falso negativo (FN) es la predicción incorrecta de una muestra positiva; siguiendo con el ejemplo, una taza que el clasificador no ha sido capaz de identificar como tal.

Falso positivo (FP) es la predicción incorrecta de una muestra negativa; por ejemplo, un objeto que NO es una taza el clasificador la identificado como una taza.

Verdadero negativo (VN) es la predicción correcta de una muestra negativa; un objeto que NO es una taza, el clasificador lo identifica como que NO es una taza.

Las entradas de la matriz contienen el número de VPs, FNs, FPs y VNs de un sistema de clasificación. Utilizando los valores contenidos en esta matriz se pueden calcular indicadores de la calidad del proceso de reconocimiento.

3.1.2. *Precision-Recall*

Estos coeficientes son medidas utilizadas para evaluar los sistemas de reconocimiento, y cuyos valores vienen determinados de la siguiente manera: entiéndase *Precision* como cuantos de los clasificados positivamente son realmente ciertos, y se calcula usando la expresión:

$$Precision = \frac{VP}{VP + FP}$$

Mientras que la medida *Recall*, o sensibilidad traducido al castellano, viene a describir la proporción de muestras positivas existentes que han sido correctamente clasificadas, y se obtiene usando la expresión:

$$Recall = \frac{VP}{VP + FN}$$

A continuación presentamos un pequeño ejemplo para poder ver como se entienden estos conceptos en el escenario desarrollado. Nuestro clasificador ha sido entrenado para distinguir entre las clases manzana, con objetos apple_1 apple_2 y pelota, con objetos ball_1 y ball_2. Asumiendo un ejemplo de 20 consultas a la base de datos, correspondientes a los siguientes objetos: 5 apple_1, 10 apple_2, 10 ball_1 y 12 ball_2, el resultado de la matriz de confusión de las pruebas es:

		Clasificado como			
		apple_1	*apple_2*	*ball_1*	*ball_2*
Realmente es	*apple_1*	4	1	0	0
	apple_2	6	3	1	0
	ball_1	0	0	7	3
	ball_2	0	0	0	12

Tabla 3.1: Matriz de confusión.

En esta matriz de confusión todas las respuestas correctas se encuentran en la diagonal de la tabla, por lo que es fácil de inspeccionar visualmente los errores cometidos por el clasificador, representados por valores distintos de cero fuera de la diagonal. En la Tabla 3.1 podemos ver que el sistema distingue bien entre ball_2 y el resto de objetos.

En el caso del objeto apple_1, de las cinco consultas sobre dicho objeto, el sistema ha clasificado cuatro correctamente y una de las muestras de manera errónea (correspondiente al elemento de la primera fila y segunda columna). Mas en detalle, la tabla de confusión, Tabla 3.2, para el objeto apple_1 es:

4 Verdaderos Positivos (4 apple_1 que fueron clasificadas como apple_1)	1 Falso negativo (1 apple_1 que fue marcada como apple_2)
6 Falsos Positivos (6 apple_2 que fueron clasificados como apple_1)	28 Verdaderos Negativos (todas las instancias restantes clasificadas como NO apple_1)

Tabla 3.2: Tabla de confusión para el objeto apple_1 y la clasificación obtenida en la Tabla 3.1

3.2. Diseño de los experimentos

3.2.1. Entorno de desarrollo

Como se mencionó en la Sección 3.1, todo el proyecto ha sido desarrollado sobre el *framework* de desarrollo en robótica ROS. También se han empleado un conjunto de herramientas y librerías, como OpenCV , OpenNI y PCL, de carácter open source y de fácil integración en ROS. El entorno empleado se muestra esquematizado en la Figura 3.2.

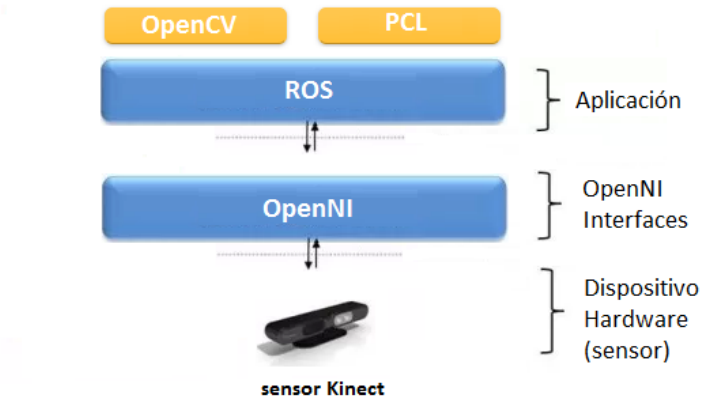


Figura 3.2: Esquema de comunicación entre el sistema y el sensor RGB-d empleado. El interfaz OpenNI simplifica la tarea al usuario, proporcionando un conjunto de funciones que hacen que todo el proceso de control de la cámara sea transparente.

Los elementos utilizados para el desarrollo de los experimentos son:

- **Sensor Kinect**
  El sensor Kinect es un dispositivo hardware que es capaz de obtener imágenes 3D comprimiendo color y profundidad. Cuenta con una cámara RGB, un sensor de infrarrojos, y micrófonos. En el Anexo A se detallan más características.

- **OpenNI (Open Natural Interaction)**
  De las diferentes librerías que existen para obtener y transmitir información del sensor Kinect se ha buscado la que ofrece una mejor compatibilidad con ROS.

  Finalmente se ha optado por utilizar OpenNI, porque tiene un gran rendimiento, y proporciona capacidades al sensor, incluyendo registro de RGB y profundidad (no requiere de una calibración previa), además de soportar diferentes resoluciones de profundidad y color. El principal objetivo de OpenNI es crear una API estándar que permita la comunicación entre los dispositivos de entrada de información y la aplicación o sistema que los recibe.

- **ROS (Robot Operating System)**

  A pesar de su nombre, ROS no es un sistema operativo propiamente dicho (de hecho funciona sobre otro sistema operativo que hace de "host", normalmente linux), sino más bien un meta-sistema o una infraestructura de desarrollo, despliegue y ejecución de sistemas robóticos, que cuenta con un gran número de repositorios que ofrecen paquetes software de todo tipo, para robots.

  ROS provee de un mecanismo de comunicaciones (*middleware*) distribuido entre nodos del sistema robótico. Entiéndase un nodo como cualquier pieza de software del sistema (desde un algoritmo SLAM hasta un driver para el manejo de un motor). Estos nodos se comunican entre ellos mediante mecanismos de paso de mensajes RPC o *Publish/Subscribe*, *Service lookup*, etc. Y permite crear arquitecturas P2P de componentes robóticos distribuidos.

  Uno de los principales motivos por los que se ha decidido implementar el sistema de clasificación utilizando ROS es su popularidad en el mundo científico. De esta manera se puede facilitar la diseminación del algoritmo de reconocimiento desarrollado en este proyecto y la posibilidad de integrarlo en otros sistemas, gracias a la modularidad que brinda ROS.

- **OpenCV (Open Source Computer Vision Library)**
  OpenCV es una librería de funciones, escritas en C/C++, para realizar tareas de procesamiento de imágenes y visión computacional, que van desde lo mas básico (acceso a píxeles individuales, despliegue en pantalla, dibujo de formas geométricas) hasta lo más avanzado (filtrado, detección de bordes, transformaciones geométricas).

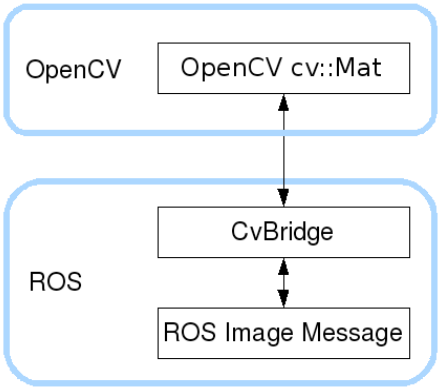


Figura 3.3: Comunicación entre ROS y OpenCV. ROS permite un fácil formateo de información.

Como muestra la Figura 3.3, ROS cuenta con mecanismos que permiten convertir la información y poder procesarla en OpenCV.

- **PCL (Point Cloud Library)**
  Point Cloud Library (PCL) es otra librería, en este caso para el procesamiento de nubes de puntos, cuyo propósito es acelerar los algoritmos 3D de percepción para el uso en aplicaciones robóticas. El framework PCL contiene numerosos algoritmos como *filtering*, *surface reconstruction*, *registration*, *model fifting* y *segmentation*, por nombrar algunos. De igual manera se ha empleado PCL debido a su fácil integración en ROS y a su gran popularidad en el procesamiento de nubes de puntos.

3.2.2. *Dataset* de imágenes

El *dataset* de imágenes que se ha empleado en este proyecto es uno de los mas referenciados a nivel mundial en la comunidad investigadora[1], el cual está formado por un gran conjunto de objetos de uso cotidiano. Los objetos se encuentran organizados en 51 categorías, que han sido dispuestas usando WordNet[2]. El conjunto de datos ha sido obtenido grabando con una cámara RGB-d Kinect, las capturas se han sincronizado y alineado a 640x480 píxeles RGB e imágenes de profundidad a 30Hz. Cada objeto cuenta con capturas de secuencia de video sobre una rotación entera, la Figura 3.4 es un ejemplo. Para cada objeto hay 3 secuencias de video, cada una grabada desde una altura diferente de modo que el objeto se ve desde diferentes ángulos

Figura 3.4: Múltiples vistas de un caja de cereales, pertenecientes a una secuencia de video.

A diferencia de otros *dataset* de imágenes existentes, como Caltech 101[3] o IMAGEnet[4], los objetos de este *dataset* se organizan en clases, objetos y lo que hemos denominado instancias (correspondientes a los *frames* de cada secuencia de video). Una clase es, por ejemplo, *lemon*, un objeto de esa clase es lemon_2, y una instancia del objeto lemon_2 es lemon_2_8. En los otros, la clase pelota, por poner un ejemplo, contiene imágenes de muchas pelotas diferentes, y no hay manera de saber si dos imágenes contienen la misma pelota, mientras que el *dataset* de imágenes RGB-d la clase pelota se divide en instancias únicas, como pelota roja o pelota amarilla. La Figura 3.5 muestra algunos ejemplos de objetos que estan incluidos en el *dataset* de imágenes.

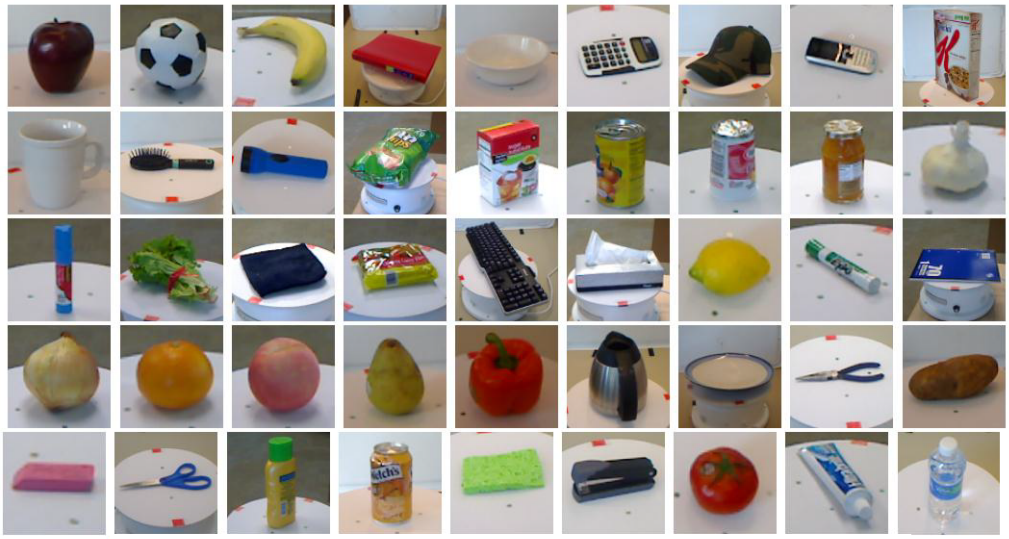


Figura 3.5: Algunos de los objetos del *dataset* de imágenes RGB-d. Cada objeto, es un ejemplo de una clase.

[1] www.cs.washington.edu/rgbd-dataset
[2] wordnet.princeton.edu
[3] www.vision.caltech.edu
[4] www.image-net.org

En la Figura 3.6 se puede ver un ejemplo de como una clase, en esta caso *apple*, está dividida en varios objetos que van desde apple_1 hasta apple_5. Cada objeto a su vez lo podemos dividir en cada uno de los *frames* de las secuencias de video, que denominamos instancias, apple_5_65. En el Anexo C se detallan todo las clases que conforman el *dataset*.

Figura 3.6: Ejemplo de una instancia de cada objeto de la clase *apple*.

3.2.3. Base de datos de características

Una vez que se ha elegido el *dataset* de imágenes, se procede a generar un modelo para cada una de las imágenes de cada instancia que van a componer la base de datos que se utilizará como referencia en el sistema de reconocimiento. Cada modelo consta de un conjunto de descriptores VFH, que sirven para describir su forma desde varios puntos de vista, un conjunto de histogramas de color y si el objeto tuviera textura, se obtendrían sus descriptores SURF. En el Anexo F se detallan los pasos de extracción de los descriptores para crear los modelos. La creación de una base de datos de características a partir de un *dataset* de imágenes, además de permitir extraer la información más representativa y útil de cada objeto, también permite reducir el tamaño que ocupa esta información. En la Tabla 3.3 podemos ver el espacio que ocupa la información proporcionada por el *dataset* de imágenes(para cada instancia de cada objeto tenemos un archivo con la nube de puntos, otro con la imagen 2D mínima, otro con la máscara mínima y otro con la profundidad); y la información que es útil para nuestro sistema, almacenada en la base de datos de características (una vez generados los modelos para cada instancia).

	Espacio en GB	Numero de ficheros
Dataset de imágenes	70	1,113,000
Base de datos de características de referencia	2,4	617,000

Tabla 3.3: Tabla comparativa de número de ficheros que conforman el *dataset* de imágenes y la base de datos de características, así como el espacio que ocupan.

3.2.4. Proceso de clasificación

3.2.4.1 Fase de entrenamiento

A partir de la base de datos de características obtenida empleando todo el *dataset* de imágenes, se ha procedido a generar otras bases de datos de características más reducidas, y que vendrán determinas por un parámetro, que hemos denominado ***step***. Este parámetro nos ha permitido realizar un muestreo de los modelos que conforman la base de datos de características en cada uno de los *training-set*.

Denominamos *training-set* a cada uno de los conjuntos que se emplean en esta fase. Estos conjuntos están formados por una base de datos de características, cuyo contenido se obtiene de la base de datos de características principal, y cuyo tamaño varía en función del parámetro anteriormente mencionado *step*. En el Anexo E se da una explicación más detallada con ejemplos.

- ***training-set 1***: Para este primer conjunto, la base de datos está formada por todos los descriptores VFH de cada instancia de cada objeto de la base de datos de características tomados de 1 en 1, es decir está formada por todos los descriptores VFH.

- ***training-set 2***: La base de datos está formada por los descriptores VFH seleccionados de 2 en 2. Es decir, se emplea únicamente solo la mitad de los descriptores VFH contenidos en la base de datos de características

- ***training-set 3***: La base de datos está formada por todos los descriptores VFH de la base datos de características tomados de 10 en 10, en este caso solo se contará con la décima parte de los descriptores. Esta reducción en el número de descriptores seleccionados se lleva a cabo buscando obtener un equilibrio entre el número de modelos empleados y la precisión del sistema.

- ***training-set 4***: La base de datos de este conjunto solo cuenta con la vigésima parte de los descriptores VFH (unos 10420 elementos). Para este último caso el número de descriptores VFH para cada objeto es de entre 30 a 40.

Los descriptores SURF e Histograma de color únicamente se cargan cuando se dispone de la lista de posibles candidatos en la fase de consulta.

3.2.4.2 Fase de consulta

Se van a realizar tres experimentos mediantes los cuales se pretende estimar el grado de respuesta del sistema de clasificación, analizando en un primer lugar su coste computacional, para a continuación evaluar la calidad de respuesta del mismo.

- **Experimento 1: Evaluación del sistema de reconocimiento utilizando escenas sencillas.**

  Se denominan escenas sencillas porque se tratan de escenas recortadas y preparadas sin apenas ruido. En la figura 3.7 se pueden ver un par de ejemplos.

Figura 3.7: Ejemplos de lo que se ha considerado escena simple. Puede observarse que apenas tienen planos que eliminar ni ruido de fondo.

  Evaluaremos nuestras técnicas en dos niveles, a nivel de categoría o clase y a nivel de objeto. El reconocimiento y detección a nivel de clase implica clasificar los objetos como pertenecientes a una misma clase. Mientras que el reconocimiento y detección a nivel de objeto consiste en identificar si un objeto es el mismo objeto que se había visto anteriormente.

  La capacidad de reconocer y detectar objetos en ambos niveles es importante si queremos utilizar este sistema de reconocimiento en el contexto de tareas robóticas aplicadas a servicios. Por ejemplo, la identificación de un objeto como un taza de café genérica o una taza de café amarilla puede tener implicaciones diferentes según el contexto.

  Para este experimento se hará uso en la fase de entrenamiento del sistema de los cuatro *traning-set*. En la fase de consulta se seleccionarán dos subconjuntos independientes. Uno que denominaremos, **subconjunto A**, en el que las escenas etiquetadas con índice par serán

las que conformen la base de datos de modelos de la fase de entrenamiento. Y otro que denominaremos,**subconjunto B**, en el que las escenas están etiquetadas con índice impar, cuyos descriptores no formarán parte de la base de datos características de la fase de entrenamiento.

Para cada subconjunto se obtendrán de forma aleatoria 10 instancias de cada objeto del *dataset* de imágenes, teniendo un total 2970 objetos a evaluar (el *dataset* de imágenes esta formado por 51 clases y 297 objetos). A la vez que se realicen estas pruebas, también se evaluará el coste computacional del sistema.

- **Experimento 2 : Evaluación y comparación de la respuesta del sistema de reconocimiento base con la del sistema de reconocimiento propuesto ante escenas sencillas (con el pre-procesado ya realizado).**

  Para el siguiente experimento se va a realizar la evaluación del sistema haciendo uso en la fase de entrenamiento de clases obtenidas del *dataset*(emplearemos los *training-set* 3 y 4, la justificación de esta elección responde a los buenos resultados obtenidos en el Experimento 1 3.3.2) mientras que la fase de consulta estará formada por un conjunto de escenas, que contienen un solo objeto, capturados mediante el sensor Kinect, y preparadas para contener la menor cantidad de ruido.

  Algunos de los objetos de las escenas sí que pertenecen a algunas de las clases que conforman el *dataset*, mientras que otros no. En la Figura 3.8 podemos ver dos objetos de la misma clase, lata de comida, pero tenemos otros casos en los que el objeto a identificar no está presente dentro del "conocimiento" del sistema, y es ahí donde radica lo interesante del experimento.

Figura 3.8: Ejemplo de objeto de *dataset* y objeto capturado.

Para este segundo test se cuenta con 32 clases a evaluar y 10 objetos por clase, lo que hacen un total de 320 objetos a evaluar.

- **Experimento 3: Evaluación y comparación de la respuesta del sistema de reconocimiento base con la del sistema de reconocimiento propuesto ante escenas cotidianas complejas.**

  Una vez evaluada la respuesta del sistema de reconocimiento, tanto el sistema de reconocimiento base como para el desarrollado, ante escenas ya recortadas y pre-procesadas; vamos a proceder a su evaluación ante escenas más complejas. Esto nos va a permitir evaluar la respuesta del sistema de reconocimiento desarrollado, del sistema de reconocimiento base y de manera secundaria obtener información sobre posibles carencias del sistema a lo largo de las distintas fases de pre-procesado de la escena. Estas escenas van desde muy simples a más complejas, añadiendo grados de dificultad como oclusión de objetos o poca separación de las superficies. En la Figura 3.9 podemos observar dos ejemplos de los distintos tipos de escenas capturados para este experimento. Se dispondrá de 100 *frames* correspondientes a 25 escenas divididas en cinco niveles en función de la complejidad de la escena, cada escena es tomada desde 4 puntos de vista distintos.

  En las escenas aparecen objetos muy similares y objetos bastante distintos, aunque siempre pertenecientes a alguna de las clases etiquetadas. En cuanto al conjunto de elementos que

componen la fase de entrenamiento, para este último experimento solo se ha empleado el *training-set* 4.

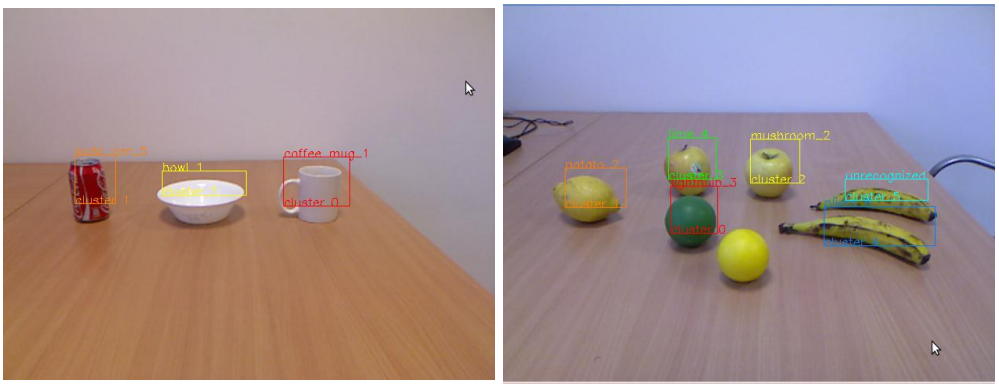


Figura 3.9: Ejemplos de escenas complejas capturadas. Cada rectángulo en las escenas indica que el sistema ha detectado un objeto e indica lo que piensa que es. En la escena de la izquierda ha identificado y clasificado de manera correcta los objetos de la escena, mientras que en la escena de la derecha no ha identificado todos los objetos, y tampoco los ha podido clasificar de manera correcta.

3.3. Análisis de resultados

3.3.1. Coste computacional

Durante la realización de los diferentes experimentos se ha tenido presente la gran cantidad de información que se ha empleado. Es por este motivo que se ha tenido que implementar una nueva manera de adquirir la información durante la fase de entrenamiento. Adquirir esta información no es otra cosa que cargar todos los descriptores de los que el sistema va ha hacer uso. Conviene recordar que la cantidad de información que el sistema va a tomar para construir su base de datos de características viene dada por el parámetro *step*. En el Anexo E se detalla la implementación del sistema propuesto que hemos realizado.

Este análisis se ha llevado a cabo con la configuración del experimento 1, para que se tenga presente los parámetros empleados. A continuación se muestran las Tablas comparativas 3.4 y 3.5 de los tiempos medios, expresados en segundos, correspondientes a las fases más significativas del sistema. En ellas se muestran los tiempos obtenidos por el sistema de reconocimiento base y por nuestro sistema de reconocimiento. En la Tabla 3.4 los tiempos para *step*=1 y *step*=2 no se han podido calcular por limitaciones del hardware (estas limitaciones no permitían mantener todos los descriptores en memoria antes incluso de poder llegar a cargarlos todos).

		Fase de Entrenamiento		Fase de Consulta		
training-set	step	*Load*	*Kd-tree*	*Query*	*Total Querys*	*Total*
1	1	> 3600	–	–	–	–
2	2	> 3600	–	–	–	–
3	10	$305,38$	$2,13$	$0,08619$	$256,01$	$563,52$
4	20	$283,11$	$1,07$	$0,07715$	$229,16$	$513,34$

Tabla 3.4: Tiempos medios de ejecución (en segundos) del sistema de reconocimiento base.

En las Tablas 3.4 y 3.5 se puede ver que el sistema de reconocimiento propuesto ha aportado un mejora significativa al rendimiento del sistema. Se puede apreciar esta mejora en el tiempo de carga, (*Load*), de los descriptores necesarios para la construcción de la estructura de búsqueda durante la fase de entrenamiento, construcción del *kd-tree*. También se observa una mejora en el tiempo medio de ejecución de cada test (*Query*), durante la fase de consulta; lo que implica una reducción

		Fase de Entrenamiento		Fase de Consulta		
training-set	step	*Load*	*Kd-tree*	*Query*	*Total Querys*	*Total*
1	1	1713, 83	21, 29	0, 10404	309, 02	2044, 14
2	2	1107, 44	10, 6	0, 08660	257, 21	1375, 25
3	10	49, 09	2, 11	0, 06189	183, 82	235, 02
4	20	24, 71	1, 13	0, 06075	180, 45	206, 29

Tabla 3.5: Tiempos medios de ejecución (en segundos) del sistema de reconocimiento propuesto.

en tiempo total de la fase de consulta (*Total Querys*). Si se suman todas estas reducciones, se consigue mejorar sustancialmente el tiempo global de ejecución.

3.3.2. Evaluación de la calidad del sistema de reconocimiento base

Experimento 1: Evaluación del sistema de reconocimiento utilizando escenas sencillas.

Con este primer experimento se ha evaluado la respuesta del sistema de clasificación, además de determinar que cantidad de información mínima, número de descriptores de cada objeto, es necesaria para poder obtener una respuesta aceptable por parte del sistema. Los resultados obtenidos para este primer experimento muestran una gran tasa de acierto, tanto a nivel de clase, como a nivel de objeto. En la siguientes Tablas 3.6 y 3.7 se muestra la *Precision* y *Recall* obtenidos por el sistema base, expresados en tanto por uno; así como los tiempos medios en el *Load*, expresados en segundos (porque es la etapa que mayor tiempo consume dentro del *workflow* del sistema).

En una primera parte dentro del experimento 1 se evalúa el sistema a nivel de clase. Como se describe anteriormente en la Sección 3.2.4, tenemos dos subconjuntos:

- Subconjunto A: Las instancias de los objetos empleados en la fase de consulta, podrán estar contenidos dentro de las instancias de los objetos que conforman la base de datos de características generada durante la fase de entrenamiento.

- Subconjunto B: Para este segundo subconjunto, las instancias de los objetos que se emplean en la fase de consulta no están contenidos dentro de las instancias de objetos que conforman la base de datos de características generada durante la fase de entrenamiento. Pero el sistema si tiene ejemplos de otras instancias de los mismos objetos.

Subconjunto A	*Precision*	*Recall*	Tiempo medio de carga de la información - *Load*
training-set 1	0, 997	0, 999	1713
training-set 2	0, 978	0, 997	1107
training-set 3	0,958	0,993	49, 089
training-set 4	0,931	0,992	24, 705

Tabla 3.6: Tabla de resultados *Precision* y *Recall* para el Subconjunto A, a nivel de clases.

Subconjunto B	*Precission*	*Recall*	Tiempo medio de carga de la información - *Load*
training-set 1	0, 997	0, 998	1713
training-set 2	0, 965	0, 989	1107
training-set 3	0,923	0,982	49, 089
training-set 4	0,886	0,976	24, 705

Tabla 3.7: Tabla de resultados *Precision* y *Recall* para el Subconjunto B, a nivel de clases.

Se puede observar en la Tablas 3.6 y 3.7 que conforme utilizamos menos información en el *training-set* los resultados empeoran, como cabe esperar, el *training-set* 1 presenta mejores resultados que el *training-set* 4. Otro hecho importante a analizar en este experimento, es que al clasificar instancias nuevas, el valor de *Precision* baja, como era esperado, pero se mantiene en unos valores bastante buenos, lo cual indica que el sistema es robusto a la hora de clasificar nuevas instancias de las clases que conoce.

De la misma forma, también se ha procedido a evaluar el sistema a nivel de objeto, al igual que a nivel de clase, se han empleado los Subconjuntos A y B:

Subconjunto A	*Precision*	*Recall*	Tiempo medio de carga de la información - *Load*
Training-set 1	0,997	0,999	1713
Training-set 2	0,968	0,997	1107
Training-set 3	0,925	0,993	49,089
Training-set 4	0,887	0,991	24,705

Tabla 3.8: Tabla de resultados *Precision* y *Recall* para el Subconjunto A, a nivel de objetos.

Subconjunto B	*Precision*	*Recall*	Tiempo medio de carga de la información - *Load*
Training-set 1	0,997	0,998	1713
Training-set 2	0,935	0,989	1107
Training-set 3	0,852	0,979	49,089
Training-set 4	0,789	0,974	24,705

Tabla 3.9: Tabla de resultados *Precision* y *Recall* para el Subconjunto B, a nivel de objetos.

Al emplear un mayor nivel de detalle para clasificar los objetos, se puede observar en las Tablas 3.8 y 3.9, un comportamiento muy similar al presentado a nivel de clases, en cuanto a la cantidad de información empleada por el sistema y la respuesta del mismo.

En las siguientes imágenes se muestran las matrices de confusión obtenidas en el *traning-set* 1. En la Figura 3.10 podemos ver las matrices de confusión para el Subconjunto A y B a nivel clase. Mientras que en la Figura 3.11 se trata de las matrices de confusión a nivel de objeto.

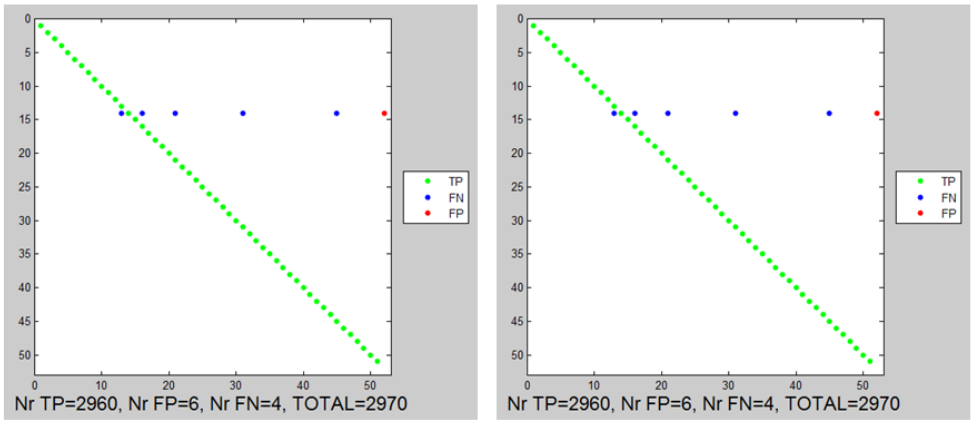


Figura 3.10: Matriz de confusión correspondiente al *training-set* 1, a nivel clases. Los puntos de color verde representan a los Verdaderos positivos (TP), en la diagonal, lo que indica que han sido clasificados correctamente. Los puntos azules son para los falsos negativos (FN) y los puntos rojos representan los falsos positivos(FP).

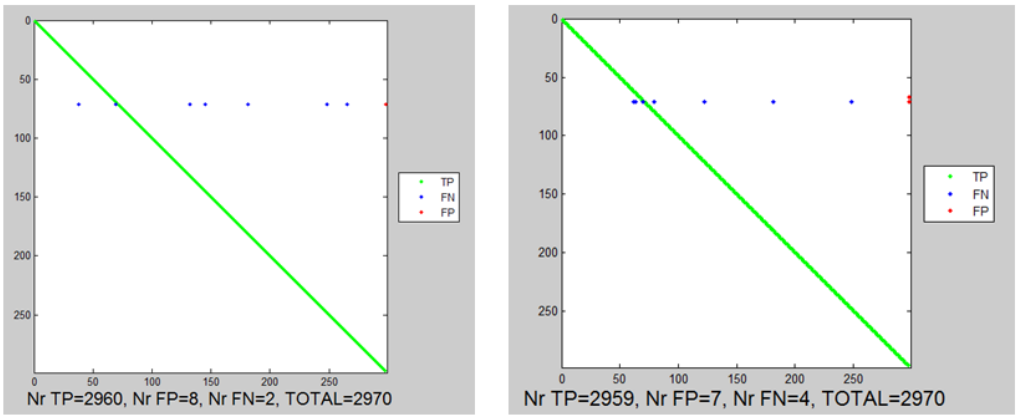


Figura 3.11: Matriz de confusión correspondiente al *training-set* 1, a nivel de objetos. La clasificación de los puntos es la misma que para la Figura 3.10.

En las Tablas anteriores podemos observar como la diferencia entre la *Precision* y el *Recall* obtenidos por los dos primeros conjuntos de pruebas (*training-set* 1, *training-set* 2) y los 2 segundos (*training-set* 3 y *training-set* 4) no presentan un diferencia muy significativa. Por el contrario, en los tiempos de ejecución si que se aprecia una gran disparidad. Se ha optado por escoger los *training-set* 3 y 4 para los siguientes experimentos. Esto es debido a nuestro compromiso entre el rendimiento y el tiempo de respuesta del sistema, ya que tanto el *training-set* 3 como *training-set* 4 presentan una respuesta muy aceptable dentro de unos tiempos razonables.

Gracias a la representación de las matrices de confusión a nivel de objeto se ha podido observar el comportamiento general del sistema de clasificación (ver Figura 3.12). Los falsos positivos (FP) en su gran mayoría quedan muy próximos a la diagonal, indicando que se trata de elementos de la misma clase (ya que los objetos de la misma clase están numerados de manera consecutiva y por lo tanto corresponden a filas-columnas contiguas). Esto es debido a que el algoritmo de decisión, penalizado por alguno de los descriptores, provoca que en vez que decidir el objeto deseado, se decida algún otro dentro de la misma clase. En el Anexo D se pueden encontrar todas las matrices de confusión y otros detalles de este experimento.

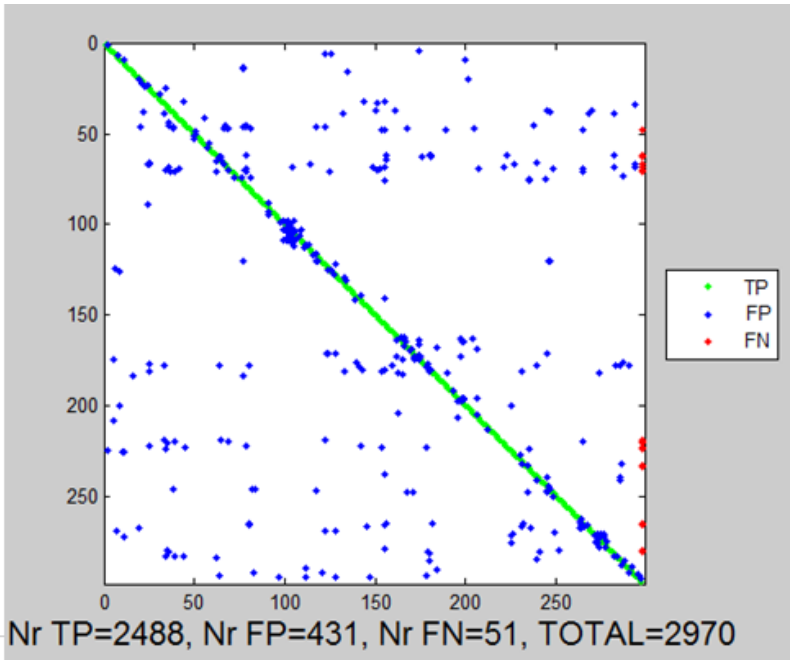


Figura 3.12: Matriz de confusión de uno de los *training-set* 3 a nivel de objeto.

Experimento 2 : Evaluación y comparación de la respuesta del sistema de reconocimiento base con la del sistema de reconocimiento propuesto ante escenas sencillas (con el pre-procesamiento ya realizado).

Para este experimento solo se ha hecho uso de los *training-set* 3 y 4, ya que como se ha observado en el experimento 1, se producen unos resultados aceptables tiempo-precisión. Como consecuencia de la evaluación de elementos que no son exactamente los mismos a los usados en la fase de entrenamiento o que el sistema no los tiene en su base de datos de características puede verse como la *Precision* y el *Recall* se reducen. En la Tabla 3.10 podemos observar los resultados obtenidos del sistema de clasificación base y del sistema de clasificación propuesto.

El objetivo de este experimento ha sido comprobar la respuesta del sistema ante objetos de los que no tiene un conocimiento previo. A partir de los resultados, mostrados en la Tabla 3.10, puede apreciarse que el sistema es sensible a la evaluación de objetos que no le hayan sido enseñados previamente, es decir, el sistema no cuenta con ningún modelo de estos objetos en su base de datos de características. Esto nos va servir como paso previo al experimento 3, en el cual se van a evaluar escenas de carácter cotidiano, analizando la influencia de la etapa de pre-procesamiento de una escena en el resultado final.

	Sistema de reconocimiento de base		Sistema de reconocimiento propuesto	
	Precision	*Recall*	*Precision*	*Recall*
Traning-set-3	0,669	0,675	0,707	0, 676
Training-set-4	0,649	0,616	0,698	0,684

Tabla 3.10: Tabla comparativa de *Precision* y *Recall* para el sistema de reconocimiento base y el sistema de reconocimiento propuesto.

Experimento 3: Evaluación y comparación de la respuesta del sistema base con la del sistema de reconocimiento propuesto, ante escenas cotidianas complejas.

Este experimento va a permitir analizar varios aspectos del sistema. El objetivo principal es analizar la respuesta de nuestro sistema de clasificación, así como la comparación de la respuesta obtenida por el sistema de reconocimiento base. De manera paralela se van a observar otros aspectos como por ejemplo si nuestro descriptor discriminador, el VFH ,realiza su función de manera correcta, además de observar posibles problemas derivados de la etapa de pre-procesamiento.

Dentro de la etapa de pre-procesamiento de la escena se va a evaluar si se realiza de manera correcta la eliminación de las superficies, y una vez realizada si es capaz de identificar los *clusters* dentro de la escena. A continuación se procede a observar si el descriptor VFH es capaz de realizar de manera correcta la función que desempeña en nuestro sistema. Es decir si es un buen discriminador, proporcionándonos una lista de posibles candidatos, entre los cuales tiene que encontrarse objeto a identificar. Y por último evaluar, la respuesta obtenida por parte de nuestro sistema de decisión y comparándola con el sistema de reconocimiento base.

Para este experimento se han capturado una serie de escenas etiquetadas en cinco niveles, del uno al cinco, dependiendo de su complejidad. En cada nivel se han etiquetado cinco escenas, y cada escena esta compuesta de 4 *frames* (en los que se captura el objeto u objetos desde diversos puntos de vista: frontal, lateral izquierdo, derecho y superior), lo que hace un total de 100 *frames*.

En la Tabla 3.11 se describe el contenido de cada uno de los niveles de dificultad de la escena, indicando el número de *frames* y el número de objeto totales en el nivel.

Nivel	Descripción	Numero de frames	Numero Total de Objetos
Nivel 1	Cinco escenas que contiene un objeto. En la Figura 3.13 podemos ver un ejemplo de un *frame* de una escena etiqueta en este nivel.	20	20
Nivel 2	Cinco escenas que contiene dos objetos En la Figura 3.13 podemos ver un ejemplo de un *frame* de una escena.	20	40
Nivel 3	Cinco escenas que contiene tres objetos. En la Figura 3.14 podemos ver un ejemplo de un *frame* de una escena.	20	60
Nivel 4	Cinco escenas que contiene tres o más objetos. En la Figura 3.14 podemos ver un ejemplo de un *frame* de una escena.	20	68
Nivel 5	Cinco escenas que contiene un objeto. En la Figura 3.15 podemos ver un ejemplo de un *frame* de una escena.	20	68

Tabla 3.11: Descripción del contenido de cada "nivel" en el que se han etiquetado las escenas de evaluación.

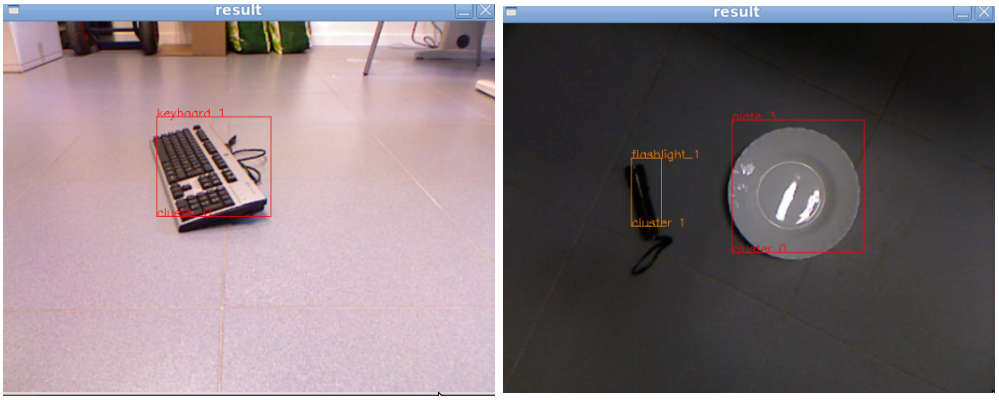


Figura 3.13: Ejemplo de escena nivel 1 (izq) y ejemplo de nivel 2 (dcha).

Figura 3.14: Ejemplo de escena nivel 3 (izq) y ejemplo de nivel 4 (dcha).

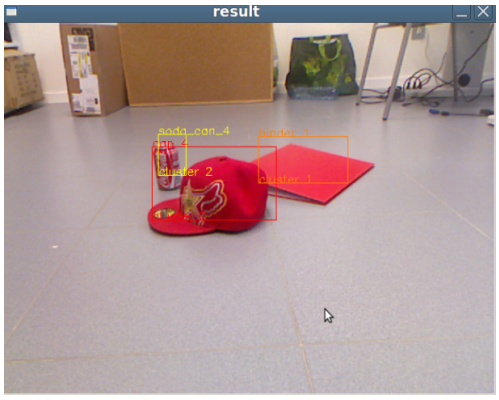


Figura 3.15: Ejemplo de escena de nivel 5.

Los resultados obtenidos del análisis de las escenas se puede observar en la Tabla 3.12. A continuación se muestra una breve descripción de lo que se ha registrado en cada columna de la tabla:

- *No Segmentados*: representa la cantidad de objetos de ese nivel que el sistema no ha sido capaz de pre-procesar correctamente.
- *Mal segmentados*: expresa que número de objetos del nivel no han sido clusterizados de manera correcta, es decir, que a la hora de analizar los *clusters*, por ejemplo, donde había dos lo ha tratado como uno.

- *No filtra VFH*: mediante esta columna se expresa la cantidad de ocasiones en las que el descriptor VFH no ha sido capaz de proporcionar una lista de candidatos en la que estuviese el objeto que realmente era, para cada uno de los objetos del nivel.
- *Sistema base*: expresa la cantidad de veces que el sistema de reconocimiento base no clasifica de manera correcta un objeto del nivel, estando éste en la lista de candidatos devuelta por el VFH.
- *Sistema propuesto*: expresa la cantidad de veces que el sistema de reconocimiento propuesto no clasifica de manera correcta un objeto del nivel, estando éste en la lista de candidatos devuelta por el VFH.

	NO segmentados	*Mal segmentados*	*No filtra VFH*	*Sistema base*	*Sistema propuesto*
Nivel 1	0 %	0 %	20 %	40 %	35 %
Nivel 2	2,5 %	0 %	32,5 %	27,5 %	25 %
Nivel 3	18,33 %	1,66 %	11,66 %	35 %	28,33 %
Nivel 4	17,64 %	2,94 %	23,52 %	29,41 %	20,58 %
Nivel 5	13,23 %	7,35 %	22,05 %	41,17 %	32,35 %

Tabla 3.12: Tabla de registro de errores cometidos por el sistema de reconocimiento base y propuesto

La tabla permite ir observando la respuesta del sistema a lo largo del todo el *workflow*, analizando así las partes mas susceptibles de producir errores, y que llevan a una mala clasificación de los objetos. Obsérvese que en los niveles 4 y 5 los porcentajes de errores a los pasos *NO segmentados*, *Mal segmentados* y *No filtra VFH*, comienzan a presentar unos valores más elevados, producidos en parte por las disposiciones de los objetos en estos niveles.

En cuanto a la respuesta de clasificación de los sistemas, la podemos ver en la siguiente tabla:

	Sistema de reconocimiento de base		Sistema de reconocimiento propuesto	
	Precision	*Recall*	*Precision*	*Recall*
Nivel 1	1	0,4	1	0,45
Nivel 2	1	0,4	1	0,425
Nivel 3	0,976	0,525	0,972	0,593
Nivel 4	0,911	0,462	0,973	0,5522
Nivel 5	0,920	0,348	0,935	0,439

Tabla 3.13: Tabla *Precision* y *Recall* obtenidos en cada uno de los niveles.

Los resultados obtenidos en esta Tabla 3.13, nos dan una idea de la robustez del sistema, que cuando identifica algo lo identifica bien. La cantidad de objetos que encuentra es escasa, indicado por un *Recall* inferior al obtenido en los otros experimentos, pero manteniendo una *Precision* alta, lo que indica que lo que ha segmentado ha sido capaz de clasificarlo correctamente.

Para finalizar, en este experimento se han podido observar una serie de comportamientos incorrectos por parte del sistema que merece la pena destacar. Al disponer de un número elevado de objetos en la base de datos, aparecen situaciones en las que los objetos son tan parecidos que es bastante complicado para el sistema poder dar una respuesta adecuada si no se han pre-procesado con precisión. A continuación se muestran dos ejemplos representativos de algunos experimentos.

- Ejemplo 1: Ejemplo demostrativo de la similitud de varios objetos del *dataset*: *calculator*, *cell_phone* y *key_board*. Como se puede ver en la Figura 3.16, las imágenes de estos objetos comparten ciertas similitudes en cuanto a forma y color.

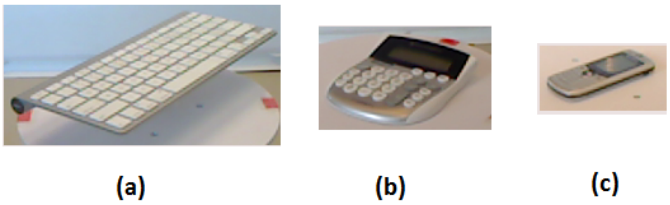


Figura 3.16: Objetos (a) *key_board*, (b) *calculator*, (c) *cell_phone*.

En la Figura 3.17 podemos ver la respuesta devuelta por el sistema identificando el objeto en la escena. Como se puede ver, se trata de un *cell_phone*, pero la respuesta de nuestro sistema de clasificación indica que tiene mayor probabilidad de ser los otros dos objetos más parecidos. La respuesta del sistema, indicando la probabilidad de que sea cada objeto:

Cluster_0:
calculator 25.5382 %
keyboard 25.123 %
binder 15.3668 %
marker 14.4349 %
cell_phone 10.1988 %
notebook 6.43956 %
hand_towel 2.89867 %

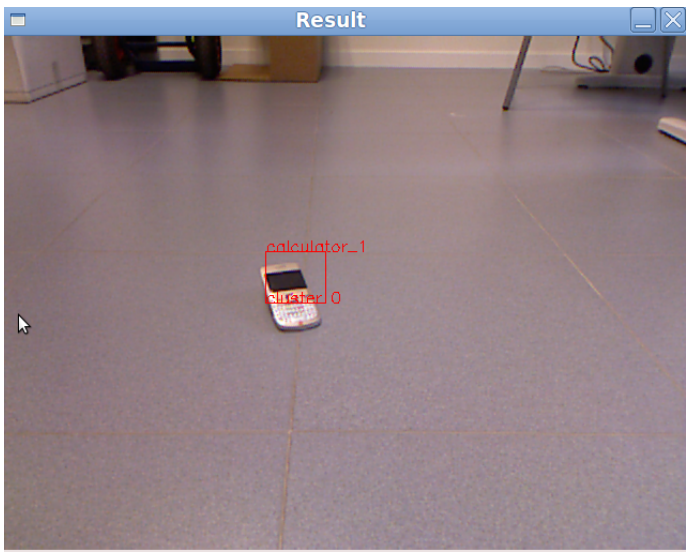


Figura 3.17: Resultado devuelto por el sistema. El sistema identifica y clasifica un objeto *cell_phone* como un objeto *calculator*.

- Ejemplo 2: Ejemplo demostrativo de los similares que son los objetos: *binder*, *notebook* y *hand_towel*. Nuevamente las imágenes de la Figura 3.18 corresponden a objetos que comparten ciertas similitudes en forma y color.

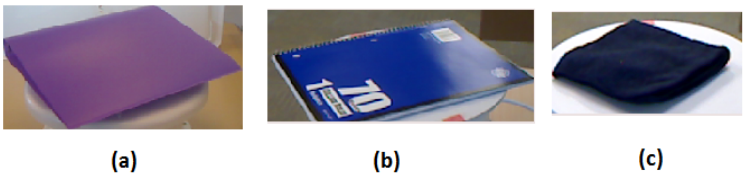


Figura 3.18: Objetos (a) *binder*, (b) *notebook* y (c) *hand_towel*.

En la Figura 3.19 podemos ver la respuesta devuelta por el sistema identificando el objeto en la escena. Como se puede ver, se trata de un *notebook*, pero la respuesta de nuestro sistema de clasificación indica que tiene mayor probabilidad de ser otro de los objetos más parecidos. La respuesta del sistema, indicando la probabilidad de que sea cada objeto:

Cluster_0:
binder 49.3833 %
notebook 16.1147 %
keyboard 12.8217 %
calculator 11.3537 %
plate 10.3267 %

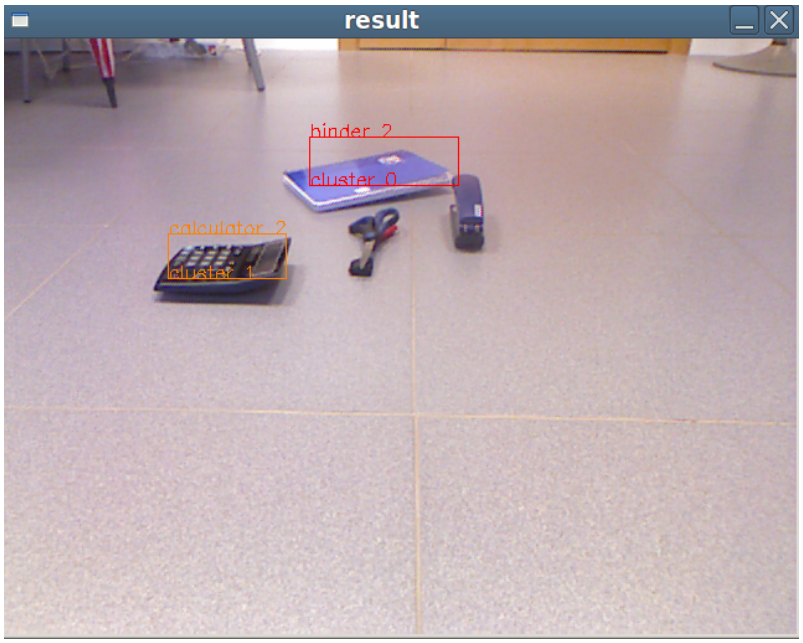


Figura 3.19: Resultado devuelto por el sistema. El sistema identifica y clasifica un objeto *notebook* como un objeto *binder*.

Los resultados obtenidos en algunas escenas generan preguntas que se plantean como propuesta de trabajos futuros en cuanto al uso de descriptores.

Capítulo 4

Conclusiones y Trabajo futuro

4.1. Conclusiones

El objetivo de este proyecto ha sido desarrollar un sistema de reconocimiento que sea capaz de identificar objetos dentro de escenas cotidianas 3D, proporcionadas por un sensor RGB-d. Ello ha implicado un estudio del estado del arte de la literatura previa, desde librerías y drivers para el manejo del sensor Kinect, pasando por las técnicas de reconocimiento que utilizan información 2D y 3D, hasta la integración de todos los requisitos funcionales bajo un misma plataforma ROS.

Se ha conseguido el objetivo general, de construir un nuevo sistema mejorado que presenta una serie de funcionalidades que mejoran los sistemas desarrollados con anterioridad. Es capaz de gestionar *dataset* de imágenes de gran tamaño de manera más eficaz y más rápida. Este objetivo se ha alcanzado mediante la reducción y mejor ordenación de la cantidad de información que el sistema necesita adquirir durante su fase de entrenamiento. Además, la nueva implementación posibilita samplear la información que el sistema necesita cargar para poder llevar a cabo su labor, llegando a un compromiso entre la precisión de respuesta del sistema y el coste computacional que supone.

También se ha mejorado y formalizado la fase de la toma de decisiones. En la fase de consulta para reconocer el contenido de una nueva imagen, se responde al usuario con una lista de candidatos priorizada, en vez de un único candidato, y estimando la probabilidad de que sea cada uno de estos candidatos, teniendo en cuenta la frecuencia de aparición de cada uno en la salida del sistema. Todas estas mejoras se han evaluado extensivamente diseñando una batería de conjuntos de test con un grado creciente de dificultad, y con ciertas características necesarias para evaluar distintas características del sistema. Se han analizado las prestaciones del sistema, así como los errores, para determinar los puntos débiles y pasos futuros donde seguir investigando..

En conjunto podemos afirmar que se han cumplidos los objetivos, presentado un sistema de reconocimiento más eficaz, que da una mejor respuesta, de manera más eficiente ante una evaluación más elaborada, además de haber formalizado la representación y evaluación del mismo.

4.2. Trabajo Futuro

Durante la realización de este proyecto han surgido ciertas cuestiones que han quedado fuera del alcance del mismo, pero que pueden ser abordadas en una investigación futura. Y que servirían para completar o mejorar el sistema de reconocimiento desarrollado.

Como posible trabajo futuro estaría emplear otro tipo de segmentación en las regiones de interés. Durante la fase experimental se ha podido apreciar que el algoritmo de segmentación básico empleado, la clusterizacion euclídea, presenta ciertas cadencias. Algunas de estas cadencias o fallos se aprecian más significativamente en escenas en las que los objetos se encuentran demasiado juntos o solapándose unos a otros. En tales casos los objetos son considerados como uno. El uso de técnicas más novedosas de pre-procesamiento podria subsanar esta serie de errores, aunque siempre habrá que tener en cuenta el balance rapidez/calidad de esta fase.

Un estudio más exhaustivo de los descriptores empleados también podría ayudar a resolver alguno de los fallos que se producen. Durante el la fase experimental se pudo observar ciertas pautas para el descriptor SURF. En ciertos objetos, el descriptor SURF obtiene suficientes puntos para que el sistema decida utilizarlo, pero sin embargo no son correctos o no tiene la precisión suficiente para obtener correspondencias entre una nueva imagen y la información de referencia. Esto puede ocurrir si se producen variaciones en los detalles (que captura el descriptor SURF). Por lo tanto, intentar utilizar más descriptores de forma global del objeto quizás ayudarían a no cometer errores por cambios en los detalles como por ejemplo los cambios de textura de unas cajas de cereales a otras, o los de una lata de cocacola a otra.

Por último, otro paso interesante sería ampliar de manera aún más si cabe el base de datos de referencia, mediante la inclusión de más objetos, así como creación un base de datos más heterogénea en cuanto a variedad dentro de cada categoría. Esto nos permitiría evaluar nuevamente si los descriptores empleados siguen siendo útiles ante esta nueva premisa o tendríamos que proceder a emplear otros.

Bibliografía

[1] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, April 2002.

[2] S. Daniel, "Side-scan sonar image matching," *The IEEE Journal of Oceanic Engineering*, vol. 23, no. 3, pp. 245–259, Jul 1998.

[3] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 364–375, Jun. 2005.

[4] A. C. Murillo, P. Campos, J. Kosecka, and J. J. Guerrero, "Gist vocabularies in omnidirectional images for appearance based mapping and localization," in *10th IEEE Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS), held with Robotics, Science and Systems*, 2010.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis Machine Intelligence.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[6] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based object labeling in 3d scenes," in *IEEE International Conference on on Robotics and Automation*, May. 2012, pp. 1330–1337.

[7] Y. Sun, L. Bo, and D. Fox, "Attribute based object identificacion," in *IEEE International Conference on Robotics and Automation*, May. 2013, pp. 2096–2103.

[8] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *IEEE International Conference on on Robotics and Automation*, May. 2011, pp. 1817–1824.

[9] P. Liang and M. I. Jordan, "An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators," in *Proceedings of the 25th International Conference on Machine Learning*, Jul. 2008, pp. 584–591.

[10] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, pp. 119–139, Dec. 1997.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Journal of Machine Learning Research.*, vol. 20, no. 3, pp. 273–297, Mar. 1995.

[12] H. Zhang, "The optimality of naive bayes," in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, 2004.

[13] D. B. Monge, "Reconocimiento de objetos en 3d utilizando sensores de vision y profundidad de bajo coste," Universidad de Zaragoza, Tech. Rep., 2012, pFC.

[14] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *IEEE International Conference on Robotics and Automation*, May. 2011, pp. 1–4.

[15] J. Rodríguez, "Metodología experimental, métodos y técnicas de minería de datos," Ph.D. dissertation, Departamento de Informática de la Universidad de Burgos, 2004.

[16] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *International Conference on Intelligent Robots and Systems (IROS) IEEE/RS*, Oct 2010, pp. 2155–2162.

[17] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department Technische Universit at Munchen Germany, 2009.

[18] A. C. Murillo, J. J. Guerrero, and C. Sagues, "Surf features for efficient robot localization with omnidirectional images," in *IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 3901–3907.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[21] B. S. Manjunath, J. R. Ohm, V. V. Vinod, , and A. Yamada, "Color and texture descriptors," *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on MPEG-7*, vol. 11, no. 6, pp. 703–715, Jun. 2001.

[22] S. Siggelkow, "Feature histograms for content-based image retrieval," Ph.D. dissertation, Albert-Ludwigs-Universitat Freiburg, 2002.

[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[24] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[25] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[26] S. Rachev, "The monge-kantorovich mass transference problem and its stochastic applications," *Theory of Probability and Its Applications*, vol. 29, no. 4, pp. 674–676, Apr. 1984.

Anexo A

Sensor RGB-d Kinect

Kinect es un controlador de juego y entretenimiento desarrollado por Microsoft para la videoconsola Xbox 360 y para PC, lanzado al mercado en Noviembre de 2010. En la Figura A.1 se muestra una fotografía del dispositivo y sus componentes.

Figura A.1: Dispositivo Kinect y sus componentes.

El sensor Kinect es una barra horizontal conectada a una pequeña base con un pivote motorizado y esta diseñado para posicionarse encima o debajo de una pantalla de vídeo. El dispositivo cuenta con una cámara RGB, un sensor de profundidad y varios micrófonos. También integra un software propietario que permite la captura de movimiento en 3D del cuerpo, reconocimiento facial y reconocimiento por voz. No obstante, en este anexo solo se van a describir las características relacionadas con la visión.

El **sensor de profundidad** consiste en un proyector de láser infrarrojo combinado con un sensor CMOS monocromático, lo que captura datos de vídeo en 3D bajo cualquier condición de luz ambiental.

La **cámara RGB** del sensor es una cámara de vídeo que ayuda al reconocimiento facial y otras funcionalidades mediante la detección de tres componentes de color: rojo, verde y azul. También proporciona vídeo a un frecuencia de 30 Hz, con una resolución de 640x480 píxeles y una profundidad de 8 bits, mientras que el sensor de profundidad cuenta con una resolución de 640x480 píxeles con una profundidad de 11 bits, lo que provee hasta 2048 niveles de sensibilidad. el sensor tiene un campo de visión angular de 57° horizontalmente y de 43° verticalmente, mientras que el pivote motorizado puede inclinar el sensor hasta 27°, hacia arriba o hacia abajo.

Desde su lanzamiento han aparecido varios controlados de código abierto que permiten integrarlo en un ordenado bajo cualquier sistema operativo, lo cual ha permitido su explotación para

fines de investigación. Actualmente este sensor es la herramienta de trabajo de multitud de investigaciones, pues sus características abren nuevas posibilidades a áreas de investigación relacionadas con la visión por computador, como pueden ser la reconstrucción de escenas 3D, la detección y reconocimiento de objetos, captura de gestos, etc.

Anexo B

Arquitectura del sistema de reconocimiento

En este anexo se presentan las diferentes partes o módulos principales de los que se compone el sistema, apartado B.1, y a continuación en el apartado B.2 se detalla el funcionamiento del sistema.

B.1. Funcionamiento del sistema

La Figura B.1 muestra el diagrama de los principales módulos del sistema desarrollado. Dichos módulos se describen a continuación:

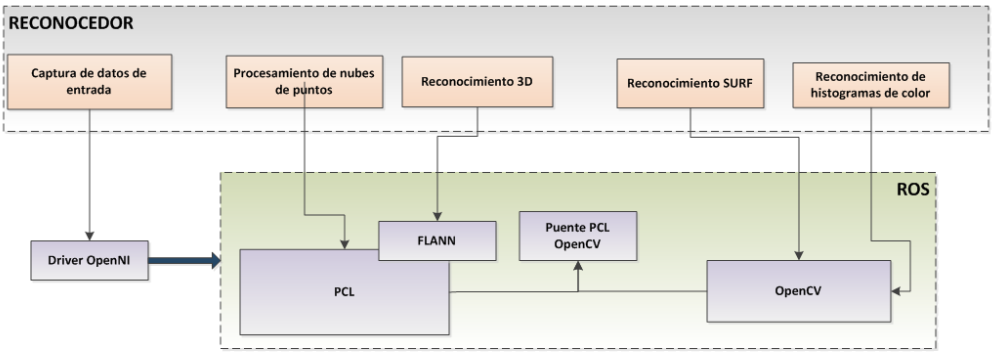


Figura B.1: Representación de los principales módulos del sistema desarrollado y su correlación con las librerías del sistema de desarrollo.

- **Reconocedor**. Es el módulo principal del sistema de reconocimiento, es como el *main* en programación. Se encarga de realizar las llamadas al resto de módulos o funciones y de decidir mediante las respuestas obtenidas cuales son los objetos a reconocer.
  - **Captura de datos de entrada**. La función de este módulo es conseguir una nube de puntos a través del sensor Kinect. El modulo utiliza el driver OpenNI para comunicarse con el sensor.
  - **Procesamiento de nubes de puntos**. Este módulo se encarga de pre-procesar las nubes de puntos de entrada. Entre sus funciones están la de eliminar el rango de visión del sensor, substraer los planos dominantes como paredes, suelo, mesas, etc y agrupar los puntos en *clusters* que representan los puntos de la nube que pertenecen a un mismo objeto. El modulo utiliza funciones implementadas en la librería PCL.

- **Reconocimiento 3D**. Este módulo es el encargado de todo lo relacionado con reconocimiento que emplea información 3D. Sus funciones son extraer los descriptores VFH de los *clusters*, compararlos con los descriptores almacenados en la base de datos de objetos a través de una búsqueda en un *kd-tree*, crear una lista de objetos candidatos y calcular una medida de similitud para cada objeto de la lista. El modulo utiliza funciones de la librería PCL para extraer los descriptores VFH y la librería FLANN para realizar la búsqueda.
- **Reconocimiento SURF**. Este módulo se encarga de todo lo relacionado con el reconocimiento por descriptores SURF. Sus funciones son extraer los descriptores SURF de la imagen mínima que engloba el *cluster* a reconocer, compararlos con los descriptores SURF de los objetos de la lista de candidatos y calcular una medida de similitud con cada objeto de la lista. El modulo utiliza funciones de la librería OpenCV para extraer los descriptores SURF.
- **Reconocimiento histogramas de color**. Este módulo se encarga de todo lo relacionado con el reconocimiento por histogramas de color. Sus funciones son calcular el histograma de color del *cluster* a reconocer, compararlo con los histogramas de los objetos de la lista de candidatos y calcular una medida de similitud con cada objeto de la lista. El modulo utiliza funciones de la librería OpenCV para calcular el espacio de color HSV a partir de la información RGB y para calcular la distancia EMD entre dos histogramas.

B.2. Funcionamiento del sistema

En este apartado se va a explicar como funciona el sistema de reconocimiento, describiendo todos los pasos que sigue hasta dar con la solución. En la Figura B.2 se describe gráficamente el proceso, y a continuación se explica con detalle los pasos del algoritmo.

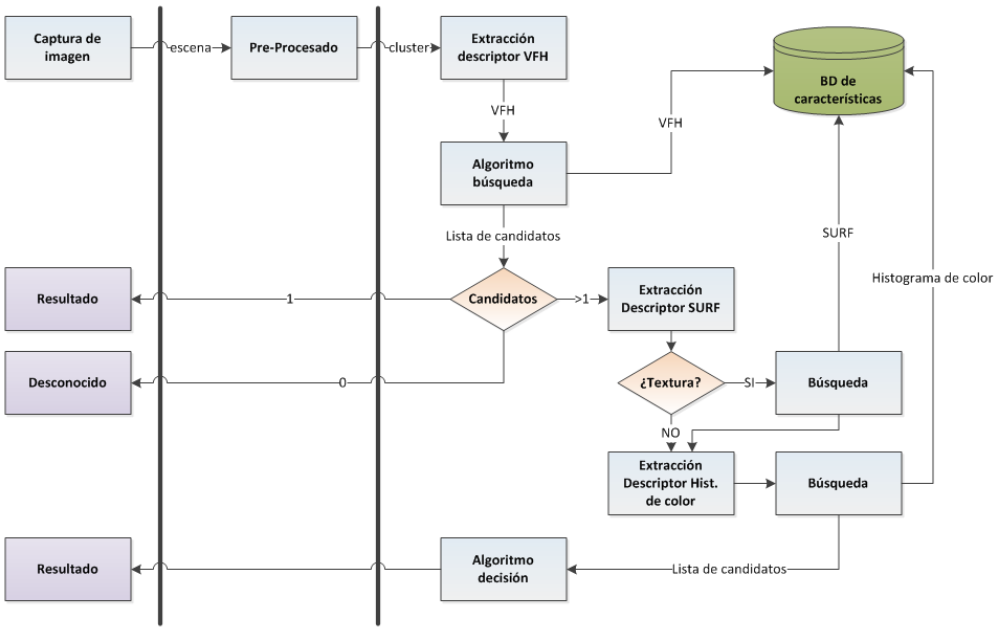


Figura B.2: Diagrama de reconocimiento del contenido de una imagen de test. Realizada la etapa de pre-procesado, para cada *cluster* que se obtiene se procede a extraer sus descriptores y se realizan las bésquedas pertinentes en busca de las similitudes con los descriptores de los objetos ya almacenados en el base de datos de características. Una vez obtenida dicha información, el algoritmo de decisión nos devolverá una lista de los posibles objetos.

La información de entrada al sistema se corresponde con una imagen(nube de puntos) capturada mediante el sensor RGB-d. Esta imagen es pre-procesada con el fin de poder definir los posibles *clusters* que representan los objetos contenidos en la imagen. Una vez identificados los *clusters*, para cada uno, se tiene la nube de puntos recortada, a partir de la cual se obtiene la imagen 2D mínima y su máscara; como en el proceso de creación de los modelos del Anexo F.

A partir de este momento, los siguientes pasos se realizan para cada *cluster* encontrado son:

- Extraer el descriptor VFH del *cluster* y realizar una búsqueda con la base de datos de descriptores VFH almacenados. Esta búsqueda se realiza en un *kd-tree* y se calcula la distancia $Chi-cuadrado$ del descriptor de test a los k vecinos más cercanos. El resultado es un lista de posibles objetos candidatos, que el sistema considera que pueden ser el *cluster* en cuestión, visto en la Sección 2.4.2.2. Esto permite al sistema realizar las siguientes búsquedas sobre un subconjunto bastante reducido de posibles objetos, lo que permite una mayor rapidez. Como se explica en el ejemplo del Anexo E, la lista de objetos candidatos puede presentar tres posibles opciones. Si la lista no contiene ningún candidato, se dice que el objeto a reconocer no corresponde con ningún objeto de la base de datos, si solamente hay un candidato, esa será la solución. Pero si la lista contiene más de un candidato, son necesarios los siguientes pasos para intentar decidir de qué objeto se trata.
- Extraer los puntos SURF de la imagen del *cluster*, con ayuda de la máscara para solo hacerlo en las zonas que pertenezcan al objeto. Si se supera un umbral determinado de puntos encontrados, se considera que el objeto tiene textura y se realiza una búsqueda, mientras que si no se alcanza el umbral se pasa al paso siguiente. Esta búsqueda está basada en el algoritmo de búsqueda del vecino más cercano, pero solo se compara con los objetos de la lista de candidatos. Por este motivo la búsqueda que se hace es exhaustiva en lugar de aproximada, pues se comparará con pocos candidatos.
- A continuación, obtener el histograma de color del *cluster* y realizar una búsqueda con los histogramas de los objetos de la lista de candidatos. Como medida de similitud se utiliza la distancia EMD explicada en 2.4.2.1.
- Finalmente tenemos una la lista de candidatos los cuales tiene asociada uno valores normalizados obtenidos de las diferentes búsquedas. Un algoritmo de decisión se encarga de estimar a partir de esa información cual es el objeto. Para más información sobre este paso del algoritmo véase la Sección 2.4.2.2.

Anexo C

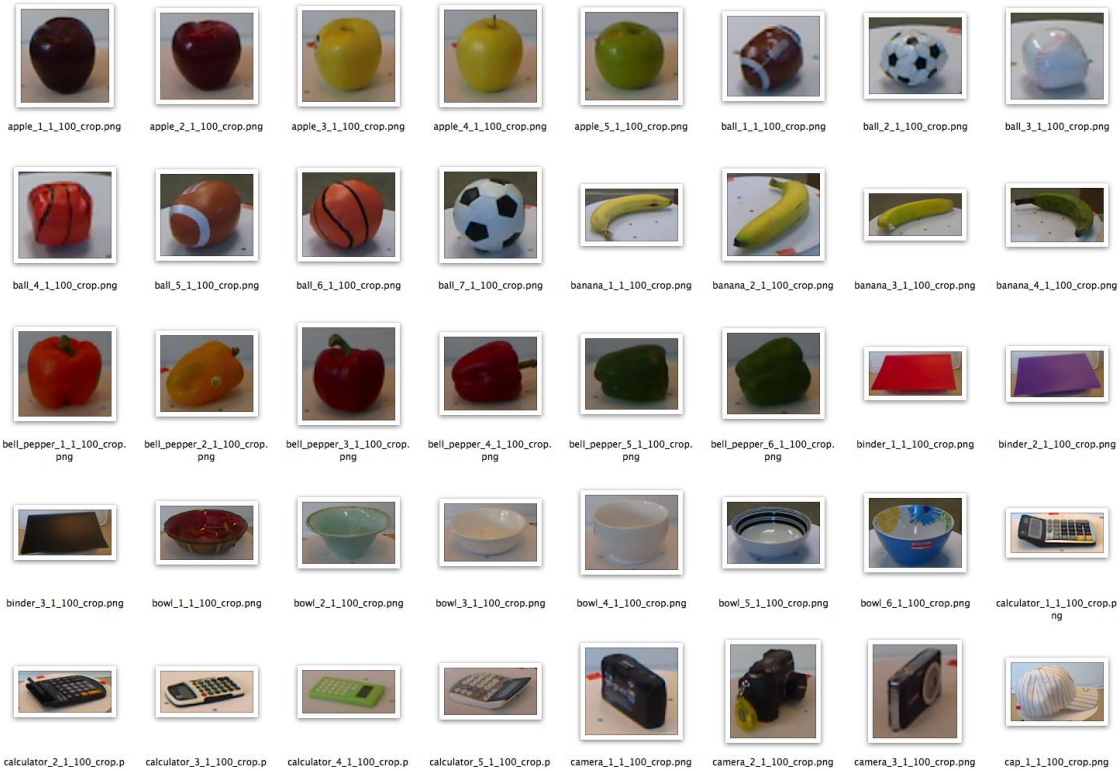
Información de los objetos del *dataset*

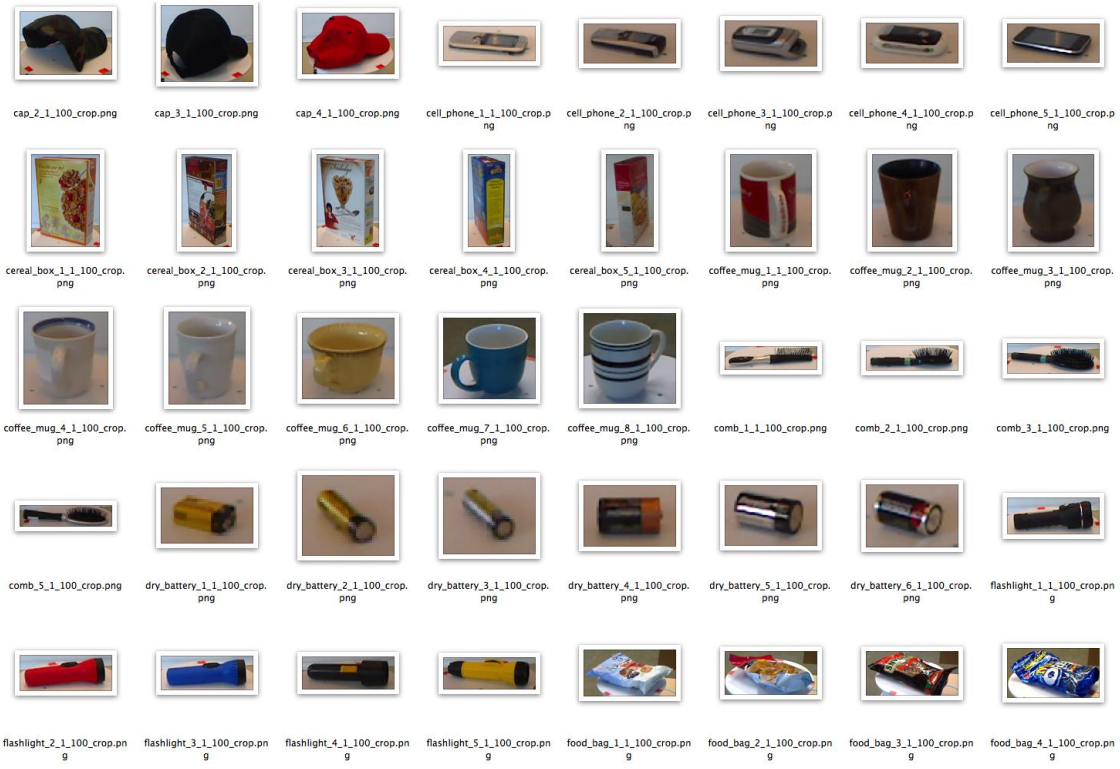
La Tabla C.1 muestra los nombres de todos los objetos empleados en la realización de los experimentos que evalúan el rendimiento del sistema. Estos objetos han sido obtenidos de un *dataset* [1] de imágenes de referencia. El *dataset* esta formado por 51 clases divididas en 297 objetos. A continuación se puede ver un ejemplo de un objeto de cada clase.

apple	ball	banana	bell_pepper	binder	bowl	calculator
camera	cap	cell_phone	cereal_box	coffe_mug	comb	dry_battery
flashlight	food_bag	food_box	food_can	food_cap	food_jar	garlic
glue_stick	greens	hand_towel	instant_noodles	keyboard	kleenex	lemon
lightbulb	lime	marker	mushroom	notebook	onion	orange
peach	pear	pitcher	plate	pliers	potato	rubber_eraser
scrissors	shampoo	soda_can	sponge	stapler	tomato	toothbrush
toothpaste	water_bottle					

Tabla C.1: Nombres de los objetos correspondientes a las clases. Cada clase esta formada por varios objetos.

[1] www.cs.washington.edu/rgbd-dataset


apple_1_1_100_crop.png
apple_2_1_100_crop.png
apple_3_1_100_crop.png
apple_4_1_100_crop.png
apple_5_1_100_crop.png
ball_1_1_100_crop.png
ball_2_1_100_crop.png
ball_3_1_100_crop.png
ball_4_1_100_crop.png
ball_5_1_100_crop.png
ball_6_1_100_crop.png
ball_7_1_100_crop.png
banana_1_1_100_crop.png
banana_2_1_100_crop.png
banana_3_1_100_crop.png
banana_4_1_100_crop.png
bell_pepper_1_1_100_crop.png
bell_pepper_2_1_100_crop.png
bell_pepper_3_1_100_crop.png
bell_pepper_4_1_100_crop.png
bell_pepper_5_1_100_crop.png
bell_pepper_6_1_100_crop.png
binder_1_1_100_crop.png
binder_2_1_100_crop.png
binder_3_1_100_crop.png
bowl_1_1_100_crop.png
bowl_2_1_100_crop.png
bowl_3_1_100_crop.png
bowl_4_1_100_crop.png
bowl_5_1_100_crop.png
bowl_6_1_100_crop.png
calculator_1_1_100_crop.png
calculator_2_1_100_crop.p
calculator_3_1_100_crop.p
calculator_4_1_100_crop.p
calculator_5_1_100_crop.p
camera_1_1_100_crop.png
camera_2_1_100_crop.png
camera_3_1_100_crop.png
cap_1_1_100_crop.png


cap_2_1_100_crop.png
cap_3_1_100_crop.png
cap_4_1_100_crop.png
cell_phone_1_1_100_crop.png
cell_phone_2_1_100_crop.png
cell_phone_3_1_100_crop.png
cell_phone_4_1_100_crop.png
cell_phone_5_1_100_crop.png
cereal_box_1_1_100_crop.png
cereal_box_2_1_100_crop.png
cereal_box_3_1_100_crop.png
cereal_box_4_1_100_crop.png
cereal_box_5_1_100_crop.png
coffee_mug_1_1_100_crop.png
coffee_mug_2_1_100_crop.png
coffee_mug_3_1_100_crop.png
coffee_mug_4_1_100_crop.png
coffee_mug_5_1_100_crop.png
coffee_mug_6_1_100_crop.png
coffee_mug_7_1_100_crop.png
coffee_mug_8_1_100_crop.png
comb_1_1_100_crop.png
comb_2_1_100_crop.png
comb_3_1_100_crop.png
comb_5_1_100_crop.png
dry_battery_1_1_100_crop.png
dry_battery_2_1_100_crop.png
dry_battery_3_1_100_crop.png
dry_battery_4_1_100_crop.png
dry_battery_5_1_100_crop.png
dry_battery_6_1_100_crop.png
flashlight_1_1_100_crop.png
flashlight_2_1_100_crop.png
flashlight_3_1_100_crop.png
flashlight_4_1_100_crop.png
flashlight_5_1_100_crop.png
food_bag_1_1_100_crop.png
food_bag_2_1_100_crop.png
food_bag_3_1_100_crop.png
food_bag_4_1_100_crop.png

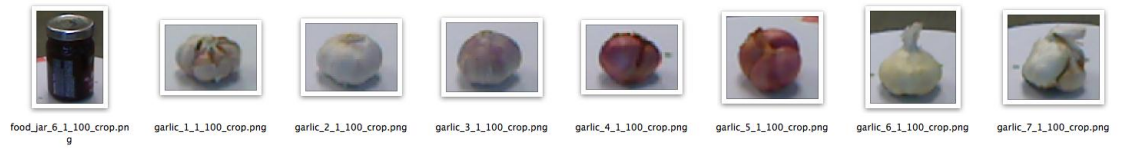
food_bag_5_1_100_crop.png food_bag_6_1_100_crop.png food_bag_7_1_100_crop.png food_bag_8_1_100_crop.png food_box_1_1_100_crop.png food_box_2_1_100_crop.png food_box_3_1_100_crop.png food_box_4_1_100_crop.png

food_box_5_1_100_crop.png food_box_6_1_100_crop.png food_box_7_1_100_crop.png food_box_8_1_100_crop.png food_box_9_1_100_crop.png food_box_10_1_100_crop.png food_box_11_1_100_crop.png food_box_12_1_100_crop.png

food_can_1_1_100_crop.png food_can_2_1_100_crop.png food_can_3_1_100_crop.png food_can_4_1_100_crop.png food_can_5_1_100_crop.png food_can_6_1_100_crop.png food_can_7_1_100_crop.png food_can_8_1_100_crop.png

food_can_9_1_100_crop.png food_can_10_1_100_crop.png food_can_11_1_100_crop.png food_can_12_1_100_crop.png food_can_13_1_100_crop.png food_can_14_1_100_crop.png food_cup_1_1_100_crop.png food_cup_2_1_100_crop.png

food_cup_3_1_100_crop.pn food_cup_4_1_100_crop.pn food_cup_5_1_100_crop.pn food_jar_1_1_100_crop.pn food_jar_2_1_100_crop.pn food_jar_3_1_100_crop.pn food_jar_4_1_100_crop.pn food_jar_5_1_100_crop.pn



food_jar_6_1_100_crop.png garlic_1_1_100_crop.png garlic_2_1_100_crop.png garlic_3_1_100_crop.png garlic_4_1_100_crop.png garlic_5_1_100_crop.png garlic_6_1_100_crop.png garlic_7_1_100_crop.png

glue_stick_1_1_100_crop.png glue_stick_2_1_100_crop.png glue_stick_3_1_100_crop.png glue_stick_4_1_100_crop.png glue_stick_5_1_100_crop.png glue_stick_6_1_100_crop.png greens_1_1_100_crop.png greens_2_1_100_crop.png

greens_3_1_100_crop.png greens_4_1_100_crop.png hand_towel_1_1_100_crop.png hand_towel_2_1_100_crop.png hand_towel_3_1_100_crop.png hand_towel_4_1_100_crop.png hand_towel_5_1_100_crop.png instant_noodles_1_1_100_crop.png

instant_noodles_2_1_100_crop.png instant_noodles_3_1_100_crop.png instant_noodles_4_1_100_crop.png instant_noodles_5_1_100_crop.png instant_noodles_6_1_100_crop.png instant_noodles_7_1_100_crop.png instant_noodles_8_1_100_crop.png keyboard_1_1_100_crop.png

keyboard_2_1_100_crop.pn keyboard_3_1_100_crop.pn keyboard_4_1_100_crop.pn keyboard_5_1_100_crop.pn kleenex_1_1_100_crop.png kleenex_2_1_100_crop.png kleenex_3_1_100_crop.png kleenex_4_1_100_crop.png


kleenex_5_1_100_crop.png lemon_1_1_100_crop.png lemon_2_1_100_crop.png lemon_3_1_100_crop.png lemon_4_1_100_crop.png lemon_5_1_100_crop.png lemon_6_1_100_crop.png lightbulb_1_1_100_crop.png


lightbulb_2_1_100_crop.png lightbulb_3_1_100_crop.png lightbulb_4_1_100_crop.png lime_1_1_100_crop.png lime_2_1_100_crop.png lime_3_1_100_crop.png lime_4_1_100_crop.png marker_1_1_100_crop.png


marker_2_1_100_crop.png marker_3_1_100_crop.png marker_4_1_100_crop.png marker_5_1_100_crop.png marker_6_1_100_crop.png marker_7_1_100_crop.png marker_8_1_100_crop.png marker_9_1_100_crop.png


mushroom_1_1_100_crop.png mushroom_2_1_100_crop.png mushroom_3_1_100_crop.png notebook_1_1_100_crop.png notebook_2_1_100_crop.png notebook_3_1_100_crop.png notebook_4_1_100_crop.png notebook_5_1_100_crop.png


onion_1_1_100_crop.png onion_2_1_100_crop.png onion_3_1_100_crop.png onion_4_1_100_crop.png onion_5_1_100_crop.png onion_6_1_100_crop.png orange_1_1_100_crop.png orange_2_1_100_crop.png


orange_3_1_100_crop.png orange_4_1_100_crop.png peach_1_1_100_crop.png peach_2_1_100_crop.png peach_3_1_100_crop.png pear_1_1_100_crop.png pear_2_1_100_crop.png pear_3_1_100_crop.png


pear_4_1_100_crop.png pear_7_1_100_crop.png pear_8_1_100_crop.png pitcher_1_1_100_crop.png pitcher_2_1_100_crop.png pitcher_3_1_100_crop.png plate_1_1_100_crop.png plate_2_1_100_crop.png

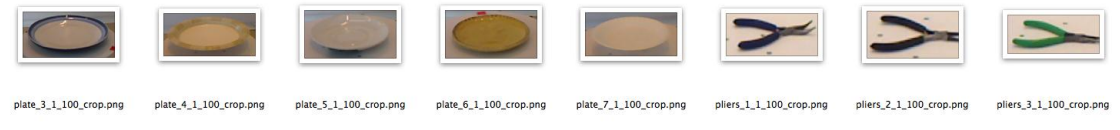
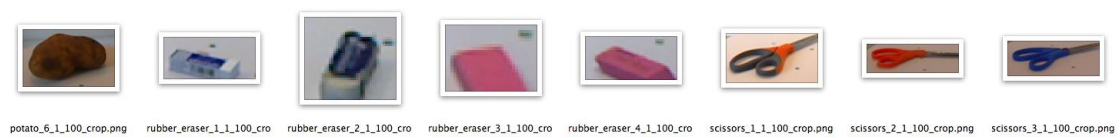


plate_3_1_100_crop.png plate_4_1_100_crop.png plate_5_1_100_crop.png plate_6_1_100_crop.png plate_7_1_100_crop.png pliers_1_1_100_crop.png pliers_2_1_100_crop.png pliers_3_1_100_crop.png


pliers_4_1_100_crop.png pliers_5_1_100_crop.png pliers_6_1_100_crop.png potato_1_1_100_crop.png potato_2_1_100_crop.png potato_3_1_100_crop.png potato_4_1_100_crop.png potato_5_1_100_crop.png


potato_6_1_100_crop.png rubber_eraser_1_1_100_cro rubber_eraser_2_1_100_cro rubber_eraser_3_1_100_cro rubber_eraser_4_1_100_cro scissors_1_1_100_crop.png scissors_2_1_100_crop.png scissors_3_1_100_crop.png


scissors_4_1_100_crop.png
shampoo_1_1_100_crop.png
shampoo_2_1_100_crop.png
shampoo_3_1_100_crop.png
shampoo_4_1_100_crop.png
shampoo_5_1_100_crop.png
shampoo_6_1_100_crop.png
soda_can_1_1_100_crop.png
soda_can_2_1_100_crop.png
soda_can_3_1_100_crop.png
soda_can_4_1_100_crop.png
soda_can_5_1_100_crop.png
soda_can_6_1_100_crop.png
sponge_1_1_100_crop.png
sponge_2_1_100_crop.png
sponge_3_1_100_crop.png
sponge_4_1_100_crop.png
sponge_5_1_100_crop.png
sponge_6_1_100_crop.png
sponge_7_1_100_crop.png
sponge_8_1_100_crop.png
sponge_9_1_100_crop.png
sponge_10_1_100_crop.png
sponge_11_1_100_crop.png
sponge_12_1_100_crop.png
stapler_1_1_100_crop.png
stapler_2_1_100_crop.png
stapler_3_1_100_crop.png
stapler_4_1_100_crop.png
stapler_5_1_100_crop.png
stapler_6_1_100_crop.png
stapler_7_1_100_crop.png
stapler_8_1_100_crop.png
tomato_1_1_100_crop.png
tomato_2_1_100_crop.png
tomato_3_1_100_crop.png
tomato_4_1_100_crop.png
tomato_5_1_100_crop.png
tomato_6_1_100_crop.png
tomato_7_1_100_crop.png
tomato_8_1_100_crop.png
toothbrush_1_1_100_crop.png
toothbrush_2_1_100_crop.png
toothbrush_3_1_100_crop.png
toothbrush_4_1_100_crop.png
toothbrush_5_1_100_crop.png
toothpaste_1_1_100_crop.png
toothpaste_2_1_100_crop.png
toothpaste_3_1_100_crop.png
toothpaste_4_1_100_crop.png
toothpaste_5_1_100_crop.png
water_bottle_1_1_100_crop.png
water_bottle_2_1_100_crop.png
water_bottle_3_1_100_crop.png
water_bottle_4_1_100_crop.png
water_bottle_5_1_100_crop.png
water_bottle_6_1_100_crop.png
water_bottle_7_1_100_crop.png
water_bottle_9_1_100_crop.png
water_bottle_10_1_100_crop.png

Anexo D

Experimentos sobre el sistema de reconocimiento 3D

En este anexo se presentan con detalle las pruebas llevadas a cabo sobre el sistema reconocimiento con el objetivo de evaluar su robustez, haciendo uso de las métricas descritas en 3.1. Estos resultados son la ampliación del Experimento 1, Sección 3.2.4.

En todas las matrices de confusión se pueden encontrar identificados los verdaderos positivos (TP) con un color verde, falsos negativos(FN) con un color azul y los falsos positivos (FP) de color rojo. Además en al pie de cada matriz se puede encontrar el numero de verdaderos positivos (TP) , de falsos negativos(FN), y de falsos positivos (FP) clasificados, así como el total de objetos evaluados.

Antes de mostrar los resultados, se va a explicar los parámetros empleados en el experimento:

Se han planteado dos subconjuntos de pruebas:

- **Subconjunto A**: las instancias de los objetos empleados en la fase de consulta, podrán estar contenidos dentro de las instancias de los objetos que conforman la base de datos de características generada durante la fase de entrenamiento.

- **Subconjunto B**: Para este segundo subconjunto, las instancias de los objetos que se emplean en la fase de consulta no están contenidos dentro de las instancias de objetos que conforman la base de datos de características generada durante la fase de entrenamiento. Pero el sistema si tiene ejemplos de otras instancias de la mismos objetos.

Se ha parametrizado la información de la que puede hacer uso el sistema, durante la fase de entrenamiento, mediante los conjuntos *training-set*:

- ***training-set 1***: Para este primer conjunto, la base de datos está formada por todos los descriptores VFH de cada instancia de cada objeto de la base de datos de características tomados de 1 en 1, es decir está formada por todos los descriptores VFH.

- ***training-set 2***: La base de datos está formada por los descriptores VFH seleccionados de 2 en 2. Es decir, se emplea únicamente solo la mitad de los descriptores contenidos en la base de datos de características.

- ***training-set 3***: La base de datos está formada por todos los descriptores VFH de la base datos de características tomados de 10 en 10, en este caso solo se contará con la décima parte de los descriptores. Esta reducción en el número de modelos seleccionados se lleva a cabo buscando obtener un equilibrio entre el número de modelos empleados y la precisión del sistema.

- ***training-set 4***: La base de datos de este conjunto solo cuenta con la vigésima parte de los descriptores VFH (unos 10420 elementos). Para este último caso el número de descriptores para cada objeto es de entre 30 a 40.

A continuación se muestran las matrices de confusión obtenidas para los subconjuntos A y B, en función del *traning-set* empleado. Para todas las Figuras la imagen de la izquierda es la respuesta obtenida para el Subconjunto A y la de la derecha para el Subconjunto B.

En este **primer conjunto de resultados** se muestra la evaluación de la respuesta del sistema de reconocimiento **a nivel de clases**.

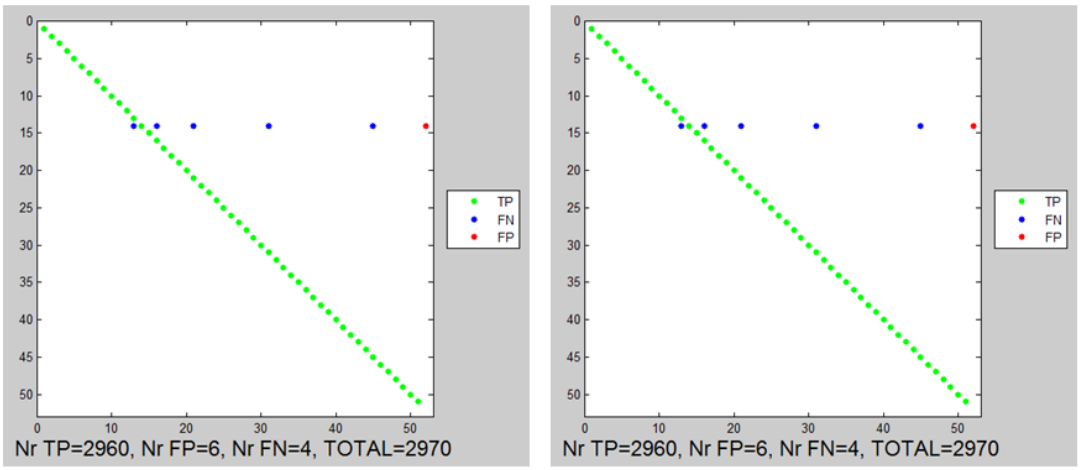


Figura D.1: Matrices de confusión correspondiente al *traning-set* 1. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

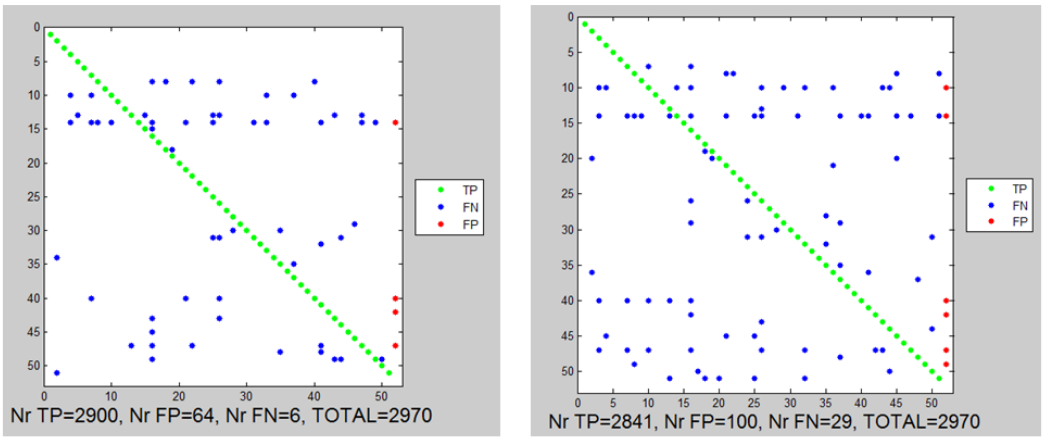


Figura D.2: Matriz de confusión correspondiente al *traning-set* 2.A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

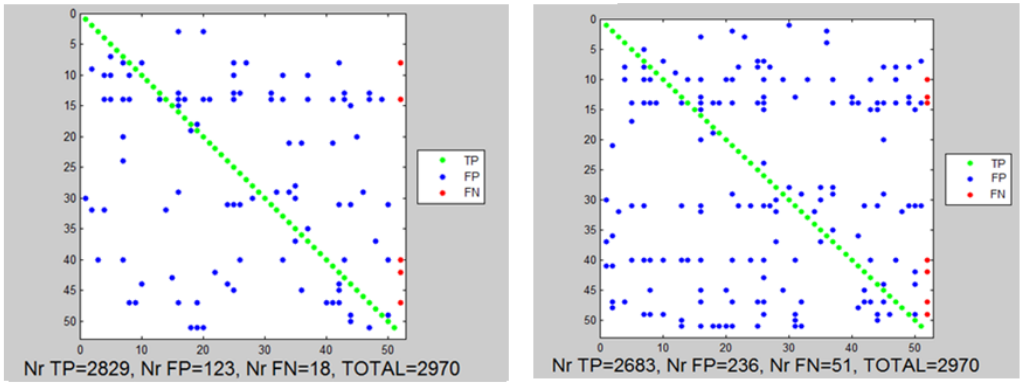


Figura D.3: Matriz de confusión correspondiente al *traning-set* 3. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

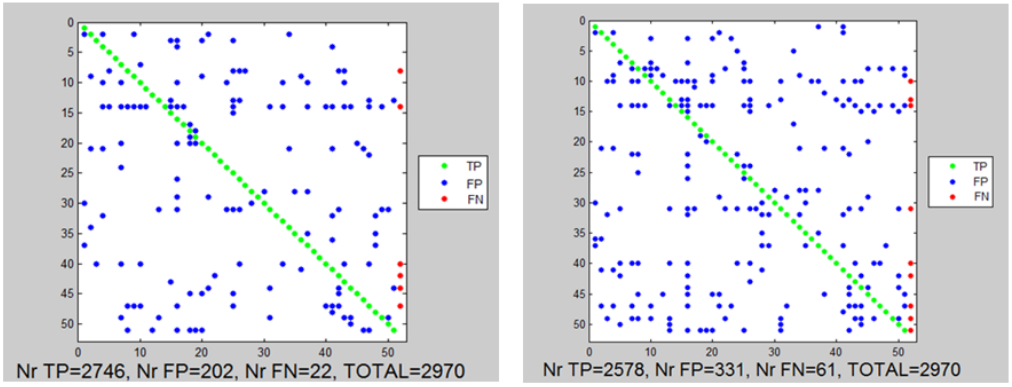


Figura D.4: Matriz de confusión correspondiente al *traning-set* 4. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

Visualmente se puede apreciar que la tasa de acierto en el Subconjunto A, es mayor que en el Subconjunto B. Esto se puede corroborar haciendo uso de los indicadores *Precision* y *Recall*. Los resultados se muestran en las siguientes Tablas de D.1.

Subconjunto A	*Precision*	*Recall*
Training-set-1	0, 997	0, 999
Training-set-2	0, 978	0, 997
Training-set-3	0,958	0,993
Training-set-4	0,931	0,992

Subconjunto B	*Precision*	*Recall*
Training-set-1	0, 997	0, 998
Training-set-2	0, 965	0, 989
Training-set-3	0,923	0,982
Training-set-4	0,886	0,976

Tabla D.1: Tablas *Precision* y *Recall* a nivel de clases, para los Subconjuntos A y B.

Los resultados obtenidos de *Precision* y *Recall*, muestran que se trata de un sistema robusto, que es capaz de encontrar bastante bien así como de identificar de manera correcta lo que ha encontrado.

En este **segundo grupo de resultados** se muestra la evaluación de la respuesta del sistema de reconocimiento **a nivel de objetos**.

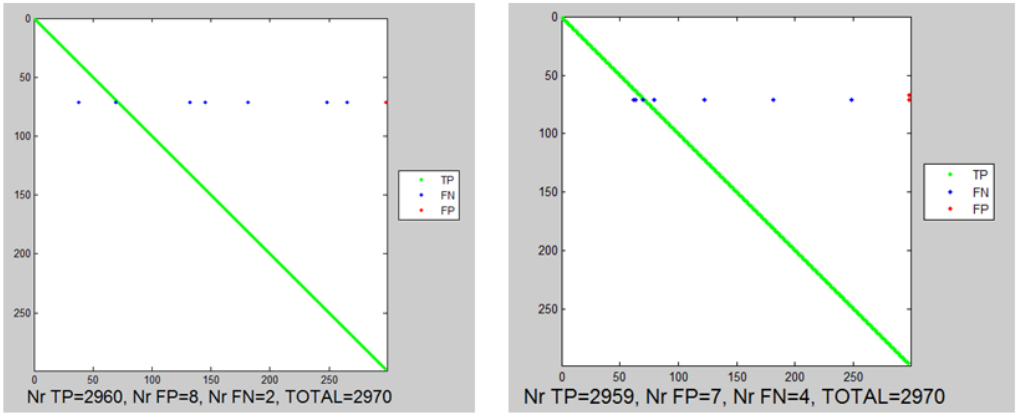


Figura D.5: Matriz de confusión correspondiente al *traning-set* 1. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

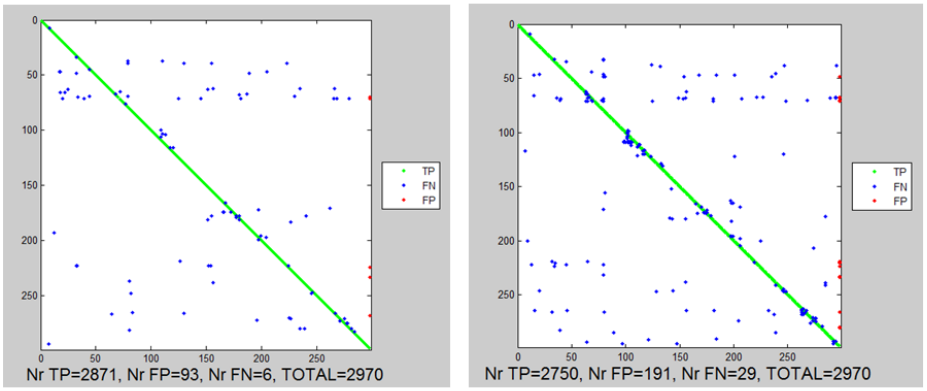


Figura D.6: Matriz de confusión correspondiente al *traning-set* 2. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

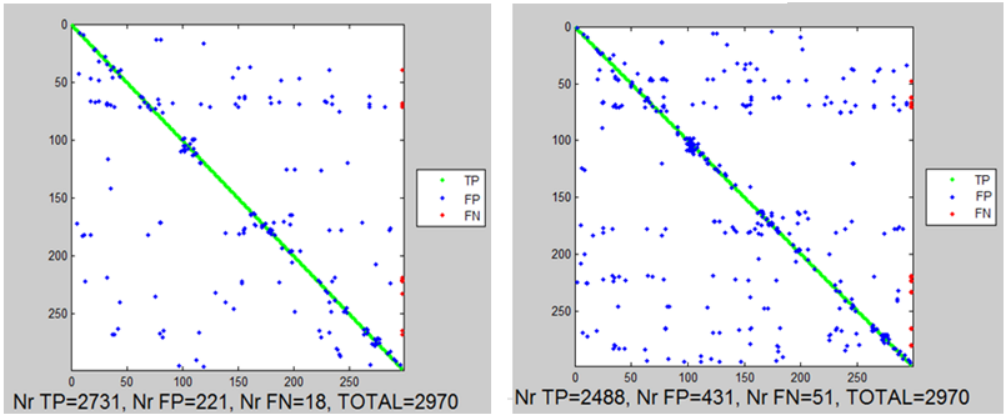


Figura D.7: Matriz de confusión correspondiente al *traning-set* 3. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

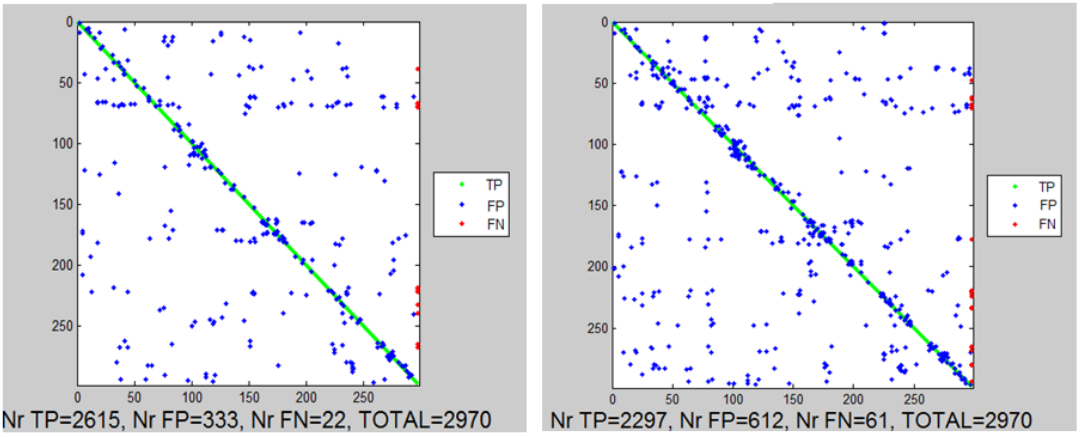


Figura D.8: Matriz de confusión correspondiente al *traning-set* 4. A la (izq) matriz correspondiente al Subconjunto A, a la (drch) matriz correspondiente al Subconjunto B.

Visualmente lo más destacable respecto a la evaluación a nivel de clases, es que al elevar el nivel de precisión a la hora de clasificar, ha implicado una mayor tasa de error en la clasificación, que no en la detección, pero como se puede ver en las Tablas de D.2, el sistema mantiene su robustez.

Subconjunto A	*Precision*	*Recall*
Training-set-1	0, 997	0, 999
Training-set-2	0, 968	0, 997
Training-set-3	0,925	0,993
Training-set-4	0,887	0,991

Subconjunto B	*Precision*	*Recall*
Training-set-1	0, 997	0, 998
Training-set-2	0, 935	0, 989
Training-set-3	0,852	0,979
Training-set-4	0,789	0,974

Tabla D.2: Tablas *Precision* y *Recall* a nivel de objetos, para los Subconjuntos A y B

Anexo E

Mejoras de rendimiento

Para poder llevar a cabo la implementación del sistema se han tenido varias cuestiones presentes, entre ellas la cantidad de información empleada, así como las limitaciones del hardware y la búsqueda de la eficiencia.

Lo que se plantea es un nuevo enfoque para gestionar la información que el sistema va adquirir durante la fase de entrenamiento. En vez de cargar (*Load*) todos los descriptores que conforman el modelo de cada instancia: descriptor VFH, SURF e Histograma de color, durante la fase de entrenamiento, para emplearlos posteriormente o no; se ha optado por cargar (*Load*) solo los descriptores VFH, empleados por el algoritmo de búsqueda 2.4.2 para construir el índice *kd-tree*. Dependiendo de la respuesta obtenida por el algoritmo de búsqueda nos encontraremos ante una de estas tres posibles opciones:

- 0 : el *cluster* a reconocer es desconocido.
- 1 : el *cluster* a reconocer es el objeto elegido.
- N : el *cluster* a reconocer puede se alguno de la lista de posibles candidatos.

Atendiendo a este resultado, opción N, se procede a cargar (*Load*) los descriptores SURF e Histograma de color de las instancias seleccionadas como posibles candidatos. De esta forma se consigue completar los modelos de las instancias que se van a necesitar en cada test (*Query*). Pero toda esta explicacion se puede ver de una manera más clara con un ejemplo.

Se captura una escena, Figura E.1, compuesta por 3 objetos: una caja de cereales, un plato y una taza.

Figura E.1: Escena capturada compuesta por objetos cotidianos.

Una vez capturada la escena, el sistema procede a cargar los descriptores VFH de todas las instancias de objetos que conforman la base de datos de características, para construir la estructura que se emplea en el algoritmo de búsqueda, el *kd-tree*. Una vez construido el *kd-tree*, se procede a la fase de pre-procesamiento de la escena, la Figura E.2 muestra el resultado de esta fase, descrita en la Sección 2.3 . Terminada esta etapa, en la escena ya solo quedan los *cluster*, que puede ser un

objeto, es en este momento donde entre en juego el algoritmo de decisión.

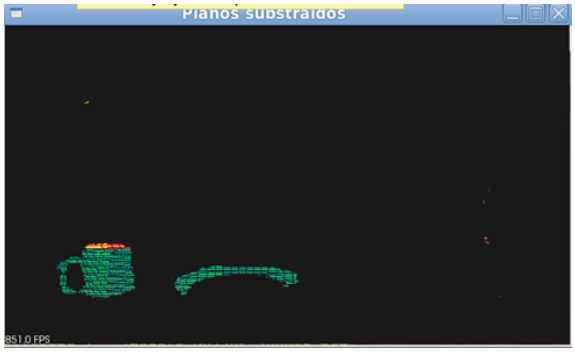


Figura E.2: Resultado del pre-procesamiento de la escena. Se pude apreciar que de los posibles *cluster*, la caja de cereales ha desparecido, posiblemente por que el sistema la ha interpretado como un plano.

Y es aquí atendiendo a la respuesta del reconocedor cuando procedemos a la segunda fase de nuestro sistema: Si el reconocedor, atendiendo al descriptor VFH, indica que no lo ha reconocido, se etiqueta como *unrecogniced*. Si lo detecta y reconoce solo uno; devuelve el objeto. Pero si devuelve una lista de posibles objetos candidatos Figura E.3, es en este punto donde se procede a cargar los descriptores SURF e Histograma de color de cada uno de los posibles instancias, necesarios para el resto de la fase de reconocimiento. De esta forma solo se mantienen el sistema aquellos modelos necesarios para cada test (*Query*).

```
Objetos posibles : 16
coffee_mug_3
coffee_mug_3_140
bell_pepper_1
bell_pepper_1_220
water_bottle_10
water_bottle_10_0
coffee_mug_1
coffee_mug_1_100
bell_pepper_5
bell_pepper_5_560
coffee_mug_5
coffee_mug_5_120
cap_1
cap_1_480
food_bag_5
food_bag_5_180
water_bottle_9
water_bottle_9_0
bell_pepper_6
bell_pepper_6_320
camera_2
camera_2_160
food_bag_8
food_bag_8_60
coffee_mug_4
coffee_mug_4_100
water_bottle_5
water_bottle_5_100
soda_can_1
soda_can_1_300
keyboard_5
keyboard_5_20
```

```
../models_DB/coffee_mug_3_140.hist
../models_DB/bell_pepper_1_220.hist
../models_DB/water_bottle_10_0.hist
../models_DB/coffee_mug_1_100.hist
../models_DB/bell_pepper_5_560.hist
../models_DB/coffee_mug_5_120.hist
../models_DB/cap_1_480.hist
../models_DB/food_bag_5_180.hist
../models_DB/water_bottle_9_0.hist
../models_DB/bell_pepper_6_320.hist
../models_DB/camera_2_160.hist
../models_DB/food_bag_8_60.hist
../models_DB/coffee_mug_4_100.hist
../models_DB/water_bottle_5_100.hist
../models_DB/soda_can_1_300.hist
../models_DB/keyboard_5_20.hist
../models_DB/coffee_mug_3_140.surf
../models_DB/bell_pepper_1_220.surf
../models_DB/water_bottle_10_0.surf
../models_DB/coffee_mug_1_100.surf
../models_DB/bell_pepper_5_560.surf
../models_DB/coffee_mug_5_120.surf
../models_DB/cap_1_480.surf
../models_DB/food_bag_5_180.surf
../models_DB/water_bottle_9_0.surf
../models_DB/bell_pepper_6_320.surf
../models_DB/camera_2_160.surf
../models_DB/food_bag_8_60.surf
../models_DB/coffee_mug_4_100.surf
../models_DB/water_bottle_5_100.surf
../models_DB/soda_can_1_300.surf
../models_DB/keyboard_5_20.surf
```

Figura E.3: Lista de los posibles objetos devueltos por el sistema (izquierda), Lista de los descriptores cargados en el sistema(derecha). El sistema solo empleará en cada *query* o consulta la información de los descriptores de la base de datos de características que necesite.

De la misma manera se repite el proceso para cada *cluster* detectado. Una vez evaluados todos los *clusters*, el reconocedor devolverá un resultado. Podemos ver en la Figura E.4 como ha identificado y reconocido la taza de café, ha identificado el *cluster* correspondiente al plato, pero no lo ha reconocido, y la caja de cereales ha sido discriminada en la fase de pre-procesamiento.

Figura E.4: Respuesta del sistema de evaluación ante la escena captura.

A continuación podemos ver una tabla comparativa Tabla E.1, que indica los tiempos medios, expresados en segundos, obtenidos para el sistema de reconocimiento base (B) y para el sistema de reconocimiento propuesto (P), en cada una de las etapas mas representativas del *workflow* del sistema.

	B	P	B	P	B	P	B	P	B	P
step	Load	Load	Kdtree	Kdtree	Query	Query	$\sum$ Querys	$\sum$ Querys	Total	Total
1	> 3600	$1713,83$	–	$21,29$	–	$0,10404$	–	$309,02$	–	$2044,14$
2	> 3600	$1107,44$	–	$10,6$	–	$0,0866$	–	$257,21$	–	$1375,25$
10	$305,38$	$49,09$	$2,13$	$2,11$	$0,08619$	$0,06189$	$256,01$	$183,82$	$563,52$	$235,02$
20	$283,11$	$24,71$	$1,07$	$1,13$	$0,07715$	$0,06075$	$229,16$	$180,45$	$513,34$	$206,29$

Tabla E.1: Tabla comparativa de los tiempos medios de cada sección para el sistema de reconocimiento base (B) y para el sistema de reconocimiento propuesto (P). Se puede apreciar claramente la mejora significativa en todas las etapas que permite el sistema propuesto.

Anexo F

Creación de modelos de los objetos a reconocer

Cada modelo de una instancia de un objeto almacenado en la base de datos de características consta de un conjunto de descriptores VFH que denotan su forma y punto de vista desde que se tomo la imagen, un conjunto de histogramas de color y si el objeto tuviese textura, descriptores SURF que lo describen. Todo este conjunto de datos proviene de las nubes de puntos tomadas para cada objeto.

La entrada al sistema de creación del modelo consta por tanto de un numero de nubes de puntos del mismo objeto. En la Figura F.2 se explica el proceso de creación de dicho modelo. El proceso comienza con la adquisición de la imagen(nube de puntos). Esta nube de puntos es pre-procesada para obtener el *cluster* correspondiente al objeto. A partir de este *cluster* se obtienen: una imagen 2D del tamaño mínimo que engloba al objeto y una imagen de la máscara que denota cuales de los píxeles de la imagen 2D pertenecen realmente al objeto, estas tres imágenes se pueden ver la Figura F.1.

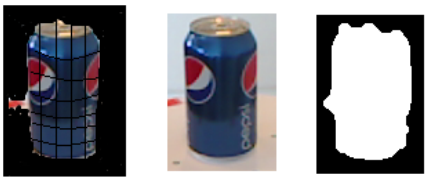


Figura F.1: Datos de la nube de puntos tras el pre-procesado. De izquierda a derecha tenemos la nube segmentada, la imagen 2D mínima y su máscara.

Una vez terminada esta etapa, se procede a extraer el descriptor VFH del *cluster* a partir de la nube de puntos recortada. Después, gracias a la imagen 2D mínima y a su máscara, se intentan extraer el descriptores SURF del objeto, almacenando únicamente los descriptores de las imágenes que alcancen un determinado numero de puntos SURF. Nótese que un objeto puede tener textura o no, dependiendo del punto de vista, por lo que el número de imágenes de las que se guardan sus descriptores SURF puede variar desde 0 (objeto sin textura) al total de instancias (objeto con mucha textura). Por ultimo, y de nuevo a partir de la imagen 2D y la mascara, se construye el histograma de color del objeto. Conviene señalar que cada descriptor almacenado en la base de datos esta asociado a una etiqueta que denota de que instancia se trata, que se almacena al mismo tiempo que el propio descriptor.

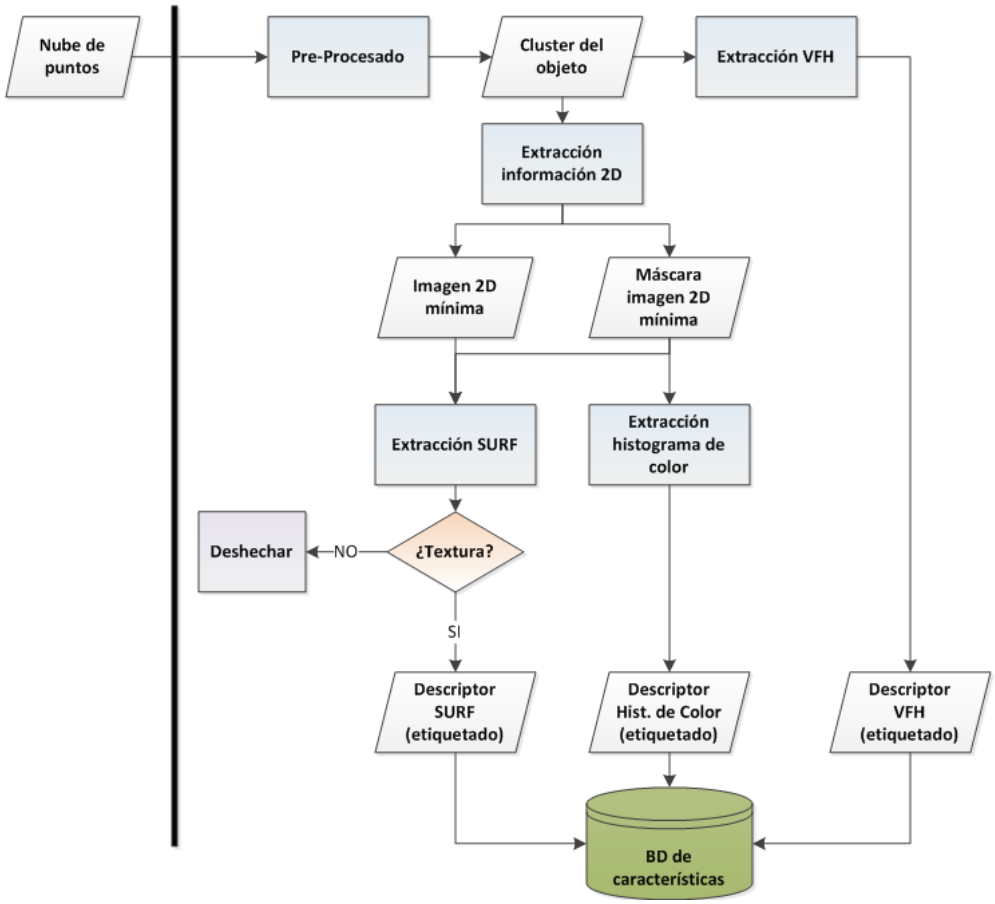


Figura F.2: Diagrama de creación de modelos de los objetos. Cada nube de puntos se pre-procesa, se extrae y almacena el descriptor VFH y a partir de cada imagen mínima y su respectiva máscara, se calcula y almacena el histograma de color y los descriptores SURF en caso de que el objeto tuviese textura. También se almacena una etiqueta junto a cada descriptor que denota de que instancia se trata.